98P2825

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Offenlegungsschrift**

(10) **DE 43 35 690 A1**

(51) Int. Cl.[5]: G 06 F 15/80

(21) Aktenzeichen: P 43 35 690.7
(22) Anmeldetag: 20. 10. 93
(43) Offenlegungstag: 4. 8. 94

DE 43 35 690 A1

(30) Innere Priorität: (32) (33) (31)
28.01.93 DE 43 02 297.9

(71) Anmelder:
Max-Planck-Gesellschaft zur Förderung der Wissenschaften e.V., 80539 München, DE

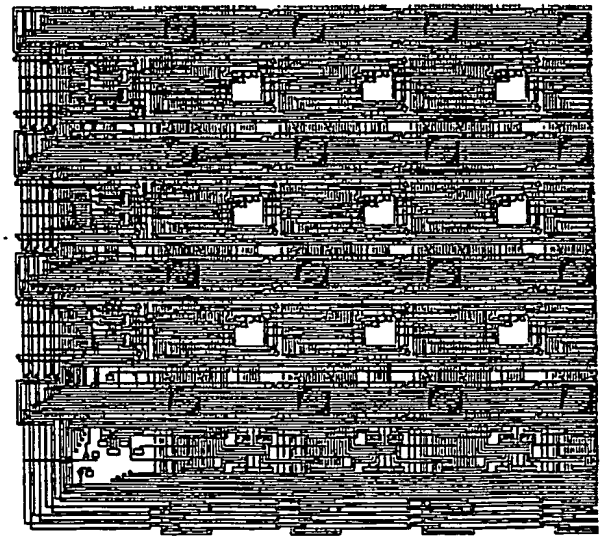
(74) Vertreter:
von Kreisler, A., Dipl.-Chem.; Selting, G., Dipl.-Ing.; Werner, H., Dipl.-Chem. Dr.rer.nat.; Fues, J., Dipl.-Chem. Dr.rer.nat.; Böckmann gen. Dallmeyer, G., Dipl.-Ing.; Hilleringmann, J., Dipl.-Ing.; Jönsson, H., Dipl.-Chem. Dr.rer.nat.; Meyers, H., Dipl.-Chem. Dr.rer.nat.; Weber, T., Dipl.-Chem. Dr.rer.nat., Pat.-Anwälte, 50667 Köln

(72) Erfinder:
McCaskill, John Simpson, 37707 Göttingen, DE

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Architektur eines und Verfahren zum Konfigurieren eines Parallelrechners

(57) Es werden eine Parallelrechnerarchitektur und ein Verfahren zum Bau und Neukonfiguration eines gewidmeten Parallelrechners für die Berechnung von Problemen beschrieben, die aus einer Vielzahl gekoppelter diskretisierbarer Vorgänge bestehen. Die besondere Rechnerarchitektur erlaubt es dem Endbenutzer, mit Hilfe eines neuen Verfahrens, wiederholt selbst einzelne diskrete Vorgänge in einem kleinen digitalen Schaltkreis zu bestimmen und danach den gesamten Rechner mit diesen gekoppelten Vorgängen zu konfigurieren, mit einem Aufwand ähnlich zur Kompilation. Weil das Problem dann direkt parallel im Hardware berechnet wird, ohne die Zwischenstufen eines Befehlssatz, können Probleme bis zu einen Millionenfach schneller als die schnellsten Workstations berechnet werden. Das Bauverfahren benutzt kommerziell erhältliche Chips und erlaubt dem Endverbraucher, einen an ihren Problemen abgestimmten massiven Parallelrechner zu bestimmen.



DE 43 35 690 A1

Beschreibung

Die Erfindung betrifft eine Architektur für einen und ein Verfahren zum Konfigurieren eines Parallelrechners.

Die vorliegende Patentanmeldung beschreibt eine Parallelrechnerarchitektur und ein Verfahren zum Bau und zur Neukonfiguration eines den Problemen eines vorbestimmten Problemkreises gewidmeten Parallelrechners für die Berechnung der Probleme dieses Problemkreises, wobei diese Probleme aus einer Vielzahl gekoppelter diskretisierbarer Vorgänge bestehen, die innerhalb des Problemkreises unterschiedlich sein können. Die besondere Rechnerarchitektur erlaubt es dem Endbenutzer, mit Hilfe eines neuen Verfahrens, wiederholt selbst einzelne diskrete Vorgänge in einem kleinen digitalen Schaltkreis zu bestimmen und danach den gesamten Rechner mit diesen gekoppelten Vorgängen zu konfigurieren, und zwar mit einem Aufwand ähnlich demjenigen bei der Kompilation. Weil das Problem dann direkt parallel "in Hardware" berechnet wird, ohne die Zwischenstufen eines Befehlssatz, können Probleme bis zu einen Millionenfach schneller als die schnellsten heutzutage erhältlichen Workstations berechnet werden. Zur Konfiguration werden kommerziell erhältliche Chips benutzt, was es dem Endverbraucher erlaubt, einen an seine Probleme abgestimmten leistungsstarken Parallelrechner zu bestimmen.

Es werden eine neuartige Computerarchitektur und ein Prototyp eines Hardware-programmierbaren parallelen (Genetik-)Prozessors zur Untersuchung von interaktiver molekularer Evolution vorgeschlagen, der derart ausgebildet ist, daß er die Simulation einer interaktiven molekularen genetischen Verarbeitung ermöglicht. Der Genetik-Prozessor erreicht die 10^9-Generationengrenze im Zeitraum von mehreren Rechnerstunden bei einer Population von 10^4 interagierenden Sequenzen variabler Länge mit einer Durchschnittslänge von 64 Bits. Der Computer ist hardware-programmierbar, wobei eine Anordnung von Field-Programable-Gate-Array-Chips (FPGA) und die mit diesen verbundenen speziellen Eigenschaften der verteilten Speicher verwendet werden. Insbesondere lassen sich die Bausteine der 4000-Familie solcher von der Firma XILINX verwenden. Die Verschiebung bei Simulationen von höheren Computersprachen zur Hardware-Ebene der Computer ist aufgrund der konzeptionellen und praktischen Vorteile vorteilhaft.

Problemstellung

Viele feinkörnige parallelisierbare Probleme der heutigen Wissenschaft, Medizin und Wirtschaft können mit der vorhandenen Kapazität der jetzigen Rechner nicht bewältigt werden. Ein typisches Beispiel ist die kombinatorische Optimierung, wie sie z. B. auf dem Gebiet der evolutionären Biotechnologie für Proteine gewünscht wird. Ein zweites Beispiel ist das Simulieren von Vielkörpersystemen, wie bei der Untersuchung von Stauverhalten im Verkehrssystem. Die konventionelle serielle von-Neumann-Maschine, wie sie in vielen modernen Rechnern wie RISC Workstations realisiert wird, ist lediglich in der Lage, sequentiell mit Hilfe eines universellen Befehlssatzes solche Probleme zu emulieren. Diese Emulation ist begrenzt durch die CPU Leistung des Prozessors, die wiederum von der endlichen Verpackungsdichte (z. B. 1µ Maskenabstände) auf Silikon Chips begrenzt ist. Es ist auch ineffizient, weil der universelle

Befehlssatz von diesen Maschinen nicht optimal an jedes Anwendungsproblem angepaßt werden kann.

Die Schwierigkeit bei der Verwendung von Parallelrechnern liegt in der Anpassung der Architektur und den Grundbefehlen des spezifischen Problems an die des Rechners: verschiedene Probleme erfordern verschiedene Parallelrechner. Es kommt hinzu, daß selbst die Kopplung mehrerer von-Neumann-Maschinen im Bereich der Synchronisierung und Datenverwaltung Probleme mit sich bringt, die sowohl bei der Herstellung von Rechnern als auch bei der Programmierung beachtlich sind. Hier wird ein Verfahren beschrieben, das es erlaubt, mit vertretbarem Aufwand einen "gewidmeten" Parallelrechner für jeden dieser Problemkreise zu bauen. Was hier angeboten wird, ist ein neues Verfahren zur Computerherstellung, eine neue Computerarchitektur und ein neues Verfahren zur Computerprogrammierung. Die patentgemäße Aufgabenstellung besteht in folgendem:

1. Eine Parallelrechnerfamilie zu entwerfen, die vom Endbenutzer in Hardware an den eigenen Problemkreis angepaßt werden kann.
2. Ein Verfahren zur Herstellung eines solchen angepaßten Rechners ohne zeitaufwendiges Chip-Design, d. h. mit bereits kommerziell erhältlichen Chips, und ohne schwierige Synchronisationsaufgaben.
3. Ein Verfahren zur Neukonfigurierung eines solchen Rechners. Das Programmieren des Rechners für ein spezifisches Problem soll direkt auf der Ebene des Konfigurierens der Hardware und nicht mittels einer Maschinensprache in Software erfolgen.

Stand der Technik

1. Verfügbare Parallelrechnerarchitekturen

Die folgende Diskussion wird auf feinkörnige Parallelrechner beschränkt, da nur bei diesen genügend Prozessoren für große Leistungsgewinne im Vergleich zu den seriellen Rechnern zur Verfügung stehen. So entfallen eine ganze Reihe von Rechnern wie Transputernetze, der Intel Hypercube, der Kendall Square (Virtual Shared Memory) Rechner usw. Während solche Rechner, mit bis zu 1000 Prozessoren, Zeitgewinne bis zu etwa einem Faktor 100 erzielen können, kommen sie nicht darüber hinaus. Entwicklungszeiten und Kosten sind enorm hoch. Es bleiben drei feinkörnige Architekturen von Rechnern, die am ehesten mit dem Erfindungsgegenstand vergleichbar sind.

a) Connection Machines (Thinking Machines Inc.)
Ursprünglich als SIMD (Single Instruction Multiple Data) Maschinen entwickelt, erlauben die Connection Machines die synchronparallele Verarbeitung bis zu 65 000 Datenträgern. Vor kurzem ist auch eine MIMD (Multiple Instruction Multiple Data) Connection Maschine auf den Markt gekommen. Die Architektur basiert in beiden Fällen und im Gegensatz zu der hier zu patentierenden Architektur auf einer klassischen Instruktionsmenge. Obwohl die Connectionmaschinen für manche feinkörnige arithmetische Berechnungen besonders gut geeignet sind, ist ihre Architektur vom Benutzer jedoch nicht beeinflußbar.
b) Cellular Automata
Ein zellulärer Automat ist eine Sammlung von iden-

tischen Automaten mit jeweils einer endlichen Zustandszahl, die regelmäßig auf einem Gitter mit ihren Nachbarn gekoppelt sind. Ein relativ allgemeiner zweidimensionaler zellulärer Automat, der direkt im Hardware programmiert werden kann, könnte auch mit dem Designverfahren hier gebaut werden. Die direkte Nachbarschaft wird in der neuen Architektur durch Verbindungskanäle mit Rutschmöglichkeiten verallgemeinert. Dadurch und mit Hilfe lokal verteilter RAM wird die Verarbeitung von Zeichenketten ermöglicht. Dieses wird im Basisdokument genauer beschrieben.

c) Ferner sind konfigurierbare Zellenanordnungen bekannt, wie z. B. in W090/11648 beschrieben. Jede Zellenanordnung ist auf einem Chip untergebracht und mehrere derartiger Chips sind in regelmäßiger (Matrix-)Form angeordnet und miteinander verbunden. Auch die Zellenanordnung innerhalb der einzelnen Chips ist matrixförmig. Derartige Anordnungen konfigurierbarer Chips werden eingesetzt, um z. B. anwenderbezogene Probleme zu lösen. Jede Zelle, jede Gruppe von Zellen oder jeder Chip dient dabei der Lösung unterschiedlicher Teilprobleme der zu berechnenden Aufgabe.

2. Herstellungsweise des Parallelrechners

Chip Design ist noch mit erheblichem Aufwand verbunden. Ein Produktions-Test-Modifikations-Zyklus dauert in der Regel mehrere Monate. Programmierbare Bausteine verkürzen zwar diesen Zyklus, sind aber wegen ihrer geringeren Verpackungsdichte und der bis jetzt starren Architektur der Rechner noch nicht für den ganzen Prozessorbau verwendet worden. In dem hier zu patentierenden Verfahren wird dagegen das Chip-Design-Problem für Parallelrechner modular gelöst. Eine über die Chipgrenze hinaus modulare Architektur beschleunigt den Designprozeß und ist so flexibel, daß Programmierer selbst innerhalb dieses Rahmens in wenigen Minuten ein für das Problem maßgeschneidertes Design implementieren können. Programmierbare Bausteine, die Festlegung eines modularen Designprinzips und die Erzeugung von problemspezifischen Chip-Designs mit Hilfe einer konventionellen Programmiersprache ermöglichen den neuen Weg.

3. Neukonfigurierungsverfahren

Ein Verfahren zur Neukonfigurierung von FPGAs wird z. B. von der Firma XILINX angeboten. Mit einem Blockschaltbildeditor werden Änderungen im Design lokal und interaktiv vorgenommen. Automatische Partition, Place and Route (PPR) Programme bestimmen das Einbetten der Logik auf dem FPGA. Dieses Verfahren hat jedoch zwei Nachteile, die es zu vermeiden gilt. Für Parallelrechner, wo das Blockschaltbild eine sich wiederholende Struktur aufweist, erfordern die sich wiederholenden Editieroperationen die Kraft einer Programmiersprache, um effizient parallele Änderungen durchzuführen.

Die gegenwärtige Forschung auf dem Gebiet der auf Genetik basierenden funktionalen Selbstorganisation erfordert einen erheblichen Datenverarbeitungsaufwand, um Populationen von interagierenden kettenkodierten (string-encoded) Automaten zu simulieren. Während die unabhängige molekulare Selektion als ein logisches Skelett wohl verstanden wird und dies insbesondere die Entwicklung evolutionärer Optimierungsalgorithmen ermöglicht hat, sind die Konsequenzen und Möglichkeiten, die sich durch interagierende Funktionalität für höhere Organisation ergeben, bisher weitgehend unverstanden. Es wurden zahlreiche Versuche, die über eine einfache Duplikation herausgingen, unternommen, um die essentielle Logik der funktionalen Interaktion zu erkennen, wobei Populationen molekularer Automaten verwendet wurden. Hier soll sowohl auf praktische als auch auf konzeptionelle Gründe für ein Abweichen von der auf diesem Gebiet herkömmlichen Software-Ausbildung und für einen Übergang zu in der Hardware realisierten Konfigurationen mit vom Anwender programmierbaren Verknüpfungsfeldern (field programmable gate arrays = FPGA) hingewiesen werden. Dies erfordert ein Umdenken in der Design-Konzeption und der Methodik, birgt jedoch auch die Aussicht auf Verbesserungen der Geschwindigkeit auf das 10^6-fache bei Populationen von bis zu 10 000 Ketten, so daß ein Teil der Flexibilität der Software-Ausbildung erhalten bleibt und konzeptionelle Vereinfachungen im Design ermöglicht werden. Ein einfaches Prototyp-Design soll im folgenden beschrieben werden, um die Nützlichkeit dieses Ansatzes und die Art der durch ihn motivierten konzeptionellen Veränderungen darzustellen.

Die einfachsten evolutiven Modelle befaßten sich mit homogenen, gut gemischten Populationen von konstanter Größe, wie sie chemisch in einem Durchfluß-Mischreaktor (continuous stirred flow reactor = CSTR) erreichbar sind. Die Selektion wird als Ergebnis der proportionalen Verdünnung zum Ausgleich des molekularen Produktionsüberschusses erreicht. In einem weiteren theoretischen Modell wurde versucht, die zur Evolution erforderlichen Elemente auf das lokal physikalisch Wesentliche zu reduzieren und die funktionalen Effekte der molekularen Interaktion einzubringen. Es erfolgte eine sukzessive Relativierung der Rolle des Zufalls als der Quelle der Vielfalt, auf die die Selektion in der Evolution einwirken kann, sowie eine Neubewertung der Bedeutung von räumlichen Effekten in der Evolution. Beide genannten Entwicklungen sind erforderlich, bevor ein Schritt hin zur FPGA-Technologie glaubwürdig sein kann.

In einem nicht-interagierenden darwinistischen Evolutionsmodell gibt es zwei Gelegenheiten, bei denen normalerweise der Zufall eine Rolle spielt: die Mutation an ausgewählten Stellen in den molekularen Sequenzen und bei der Auswahl von Molekülen, die in der Population weiter bestehen sollen. Bei interagierenden Modellen existiert ferner die zufällige Auswahl der aufeinanderfolgenden Interaktions-Partner (ein bimolekularer Prozeß, der im allgemeinen ausreicht, um biochemische Reaktionen zu beschreiben). Im letzteren Fall hat es sich als möglich erwiesen, bei kettenkodierter Funktionalität, Variationen implizit durch die unterschiedlichen Prozesse der bimolekularen Interaktion zu erzielen. Ein dynamischer Mischprozeß in einem offenen Gefäß ist als physikalische Basis für die Auswahl von Molekülen und deren Interaktionen ausreichend. In der Physik ist heute allgemein bekannt, daß einfache deterministische nicht-lineare Systeme chaotische Dynamiken erzeugen können, und es würde insbesondere ein Hartkugelmodell ausreichen, um eine lokale und selektive Randomisierung von angebrachten Ketten darzustellen. Es würde daher eine rein lokale deterministische dynamische Regel zur Beschreibung des evolutionären Vorgangs genügen. Dies hätte ferner zur Folge, daß räumlich heterogene Phänomene erstellt werden könnten. Tatsäch-

lich wurde für eine breite Gruppe von homogenen, sich entwickelnden, replizierenden Populationen festgestellt, daß sämtliche kooperativen funktionalen Interaktionen sich gegenüber parasitäreren Ausbeuten als instabil herausgestellt haben, und die Frage der räumlichen Heterogenität ist daher für jegliche Modelluntersuchung kritisch.

Die bisherigen Modellerstellungsarbeiten haben zu einer Vielzahl kleiner, gruppenweise interagierender Automaten in Computersprachen wie C geführt. Die miteinander verbundenen Erfordernisse bezüglich geeigneter Populationsgrößen (>1000) und Generationszahlen (>1000) zur Verdeutlichung von generischem Verhalten machen solche Simulationen in herkömmlichem Computerequipment zeitaufwendig. Üblicherweise benötigt die Simulation von 1000 Ketten für 1000 Generationen bei einer RISC-Arbeitsstation mehrere Stunden Rechnerzeit. Es besteht hier eine geringe Effizienz in der Verwendung von Computerbetriebsmitteln. Wie zuvor erörtert, scheint es, daß Gleitpunktoperationen zur Simulation der Evolution nicht unbedingt erforderlich sind. Zweitens sind die Prozessoren für segmentierte Operationen, die Byte- oder Wortbegrenzungen beinhalten, optimiert, nicht jedoch für Ketten variabler Länge. Drittens bestehen softwarebedingte und finanzielle Schranken für die effektive Nutzung paralleler Maschinen. Viertens sind die höheren Programmiersprachen sowohl für serielle als auch für parallele Maschinen dahingehend eingeschränkt, daß sie eine größere Breite an Interaktionsschemata zulassen als für eine genetische Verarbeitung relevant sein können.

Der Erfindung liegt die Aufgabe zugrunde, eine Vorrichtung zur Berechnung feinkörniger parallelisierbarer Probleme zu schaffen und ein Verfahren zum Konfigurieren einer derartigen Vorrichtung anzugeben, bei der der Benutzer die Vorrichtung innerhalb einer kurzen Entwicklungszeit für das zu lösende Problem konfigurieren kann und die Berechnung des Problems mit der derartig konfigurierten Vorrichtung extrem geringe Rechenzeit benötigt.

Zur Lösung dieser Aufgabe werden mit der Erfindung eine Vorrichtung mit den Merkmalen der Ansprüche 1 bzw. 2 sowie ein Verfahren mit den Verfahrensschritten des Anspruchs 3 vorgeschlagen. Die Merkmale vorteilhafter Ausgestaltungen ergeben sich aus den Unteransprüchen. Bevorzugte Verwendungen sind Gegenstand der Verwendungsansprüche.

Sinngemäß macht die Erfindung von der Erkenntnis Gebrauch, einen Rechner hardwaremäßig derart teil- oder vorzukonfigurieren, daß der Anwender schnell und einfach den Rechner für ein bestimmtes Problem einer Anzahl von Problemen (Problemkreis) optimal konfigurieren kann. Der Rechner ist also hardwaremäßig ausgelegt, um sämtliche Probleme eines vorgegebenen Problemkreises lösen zu können. Für die mit dem Rechner bzw. der Vorrichtung zu lösenden Probleme gilt, daß sie feinkörnig und parallelisierbar sind sowie als eine Vielzahl miteinander verkoppelter diskretisierbarer Prozesse beschreibbar sind.

Vorzugsweise handelt es sich bei den einzelnen Logikschaltungen um sogenannter anwenderspezifische Logikfelder (Field-Programable-Gate-Arrays — FPGA), die aus in Arrayform angeordneten einzelnen Logikzellen bestehen. Diese Logikzellen sind frei programmierbar, womit das gesamte Array frei konfigurierbar ist. Mehrere Logikzellen werden zu einzelnen Logikschaltungsblöcken, den sogenannten konfigurierbaren Logikblöcken oder CLB (Configurable Logic Block), zusammengefaßt.

Desweiteren ist erfindungsgemäß die synchrone Taktung sämtlicher Eingänge sämtlicher Logikzellen sämtlicher Logikschaltungsblöcke vorgesehen. Damit werden die bei großen Prozessornetzen mitunter auftretenden Timing-Probleme gelöst. Die FPGA verfügen über interne Verbindungsstrukturen, die es erlauben, daß das an den Eingang des FPGA angelegte Taktsignal zeitgleich an sämtlichen Logikzellen anliegt.

Ein weiteres Merkmal der Erfindung besteht darin, daß die Verbindungsmuster zwischen benachbarten CLB einer Logikschaltung (eines FPGA) regelmäßig und untereinander gleich sind. Werden mehrere FPGA verwendet, so setzt sich das benachbarte CLB verbindende Verbindungsmuster über die Grenzen der FPGA fort. D.h., daß CLB benachbarter FPGA genauso verbunden sind, wie die CLB ein und desselben FPGA.

Die Erfindung basiert unter anderem auf der Erkenntnis, verteilte Dekorrelation in vom Benutzer konfigurierbaren Logikschaltungen (nämlich den FPGA's) für die Großsimulation von Prozessen zu verwenden, die Zufallselemente erfordern. Verteilte Dekorrelation meint die Verwendung nicht korrelierter Bits von funktional getrennten aber benachbarten Digital-Schaltungen auf einem Chip zur Erzeugung quasi-zufälliger Signale an jeder Stelle, um damit stochastische Prozesse zu simulieren. Da diese Signale rekonfigurierbar sind, kann ihre Auswahl zwecks Ausräumung statistischer Vorgaben (statistical bias) optimiert werden.

Ferner werden erfindungsgemäß verteilte RAM-Speicher mit paralleler Adressierung als Schieberegister zwischen Prozessoren eingesetzt, um die Simulation größerer Ansammlungen von Objekten zu ermöglichen. Die Verwendung kürzerer RAM's mit heterogener Adressierung erlaubt lokale Einfügungen und Streichungen bei der String-Verarbeitung. Die Verwendung fest gekoppelter synchroner Ein-Ausgangsdatenströme ermöglicht es, ein "once-off-Design-Prinzip" bei allen anwendungsspezifischen Verwendungen von Speichern einzusetzen. Die Speicher können auf den FPGA's selbst verteilt sein und/oder auf dazu bestimmten Speicherchips zwischen den FPGA's verteilt sein.

Schließlich ist auch das Verbinden aller Daten- und Steuerbits in Schieberegistern zum (busbreiten) seriellen nicht-destruktiven Auslesen des augenblicklichen Zustandes einer Berechnung oder Simulation möglich. Der Datenfluß kann auf einen einfachen regulären Verschiebungsprozeß mit Defold-Voreinstellung der Prozessoren zum Auslesen reduziert werden. Der Zustand der Prozessoren selbst, was lediglich einige wenige Bits erfordert, kann in lokalen Registern festgehalten werden, und zwar vor dem Auslesen, um für spätere Speicherungen oder Einbindungen in Flipflop-Ketten zwecks Auslesens und Wiederspeicherns zur Verfügung zu stehen. Diese leistungsfähigen Eigenschaften des Auslesens werden durch die Designregeln der synchronen Eingangs -Ausgangskanäle möglich.

Schließlich läßt sich auch die Erfindung zum lokalen Rerouting im Designerstellungsprozeß für modulare Designs zwecks drastischer Erhöhung der Designimplementationszyklen bei gleicher Zeitdauer einer konventionellen Software (Editieren, Compilieren, Verbinden, Laufen) einsetzen.

Mit der Erfindung wird der Interaktions-Engpaß vermieden, indem ein neuartiges Hardwaredesign mit sich kontinuierlich schnell bewegenden Daten und rein lokaler Interaktion verwendet wird. Es werden zur Illustration ein zweidimensionaler Raum sowie binäre Ketten

verwendet, bei denen die genetischen Sequenzen auf natürliche Weise sowohl eine nahezu zufällige lokale mischende, als auch eine nahezu deterministische interaktive kopierende Funktionalität kodieren, ohne stochastische Prozesse einzubringen. Im Grunde kollidieren molekulare Ketten an einer Anordnung von binären Schaltpunkten, die deren Durchgang entsprechend dem gegenwärtig nächsten Kettennachbarn steuern.

Nachfolgend werden die erfindungsgemäße Vorrichtung und das erfindungsgemäße Verfahren genauer beschrieben.

1. Rechnerarchitektur

Auf eine Instruktionsmenge wird verzichtet. Statt dessen ist der Rechner in eine Vielzahl kleiner Prozessoren unterteilt, die aufgrund der Verwendung von in Hardware programmierbaren Bausteinen selbst völlig neu konfigurierbar sind. Jeder derartige Prozessor ist über eine begrenzte, aber sonst frei wählbare Anzahl von Bit-breiten Leitungen mit benachbarten Prozessoren verbunden. Die Nachbarschaft ist zunächst die eines zweidimensionalen Gitters. Abhängig von dem Anwendungsproblemkreis sind aber auch andere Geometrien vorgesehen. Jeder solcher Prozessor verarbeitet synchron die Bitströme auf seinen Eingangsleitungen und erzeugt ununterbrochen synchrone Bitströme auf seinen Ausgangsleitungen. Bei jedem Prozessor wird intern jeder Eingangsbitstrom vor der Weiterverarbeitung in speziellen lokalen Schieberegistern zwischengespeichert, die auch einem "random access" Zugriff erlauben. Die Prozessierung kann dabei von einer relativ langen Strecke der eingehenden Bitströme abhängen.

Die Verbindungsstruktur von Eingangs- und Ausgangsleitungen zwischen Prozessoren ist ebenfalls nicht von vornherein festgelegt, sondern lediglich eine begrenzende Obermenge der für den Problemkreis geforderten Verbindungen. Die Verbindungsstruktur zwischen Prozessoren innerhalb eines Chips wird fortgesetzt zwischen Prozessoren, die sich an nahestehenden Seiten unterschiedlicher Chips befinden. Für die Verbindung von Prozessoren, die sich nicht auf demselben Chip befinden, müssen nicht wieder konfigurierbare Leitungen gelegt werden. Die vorhandenen festen Leitungen müßten alle im Problemkreis zu realisierenden Verbindungsmuster zwischen Prozessoren erlauben. Dieses ist möglich durch programmierbare Bausteine, solange eine Obermenge der gewünschten Verbindungen fest verdrahtet ist.

Ein Bruchteil der gesamten Ein- und Ausgänge wird zusätzlich auf einem (z. B. 32 oder 64 Bit breiten) Datenbus eines seriellen oder nur grobkörnig parallelen Hostrechners geführt. Die Taktfrequenz des DMA (Direct Memory Access) Schreib-Lese Zyklus dieses Hostrechners bestimmt die Frequenz des synchron getakteten Parallelrechners. Eine Taktfrequenz um 25MHz ist mit jetziger Technologie durchaus erreichbar. Eine Kopplung mit Graphikprozessoren ist zur direkten Visualisierung auch möglich.

In den für den Prototyp benutzten Field Progamable Gate Arrays (FPGAs) muß die Architektur das Herstellen des Anfangszustandes des verteilten Speichers erlauben. Hier wird eine Defaultschaltung für die Prozessoren mit einem gesonderten Reset-Schalter aktiviert, die die Prozessoren in einfache Schieberegister umwandelt. Der aktuelle Stand des Rechners kann auf gleiche Weise ausgelesen werden. Die Zustände aller interner Flip-flops (Register) können sowohl beim Start einer Berechnung bestimmt werden als auch zwischendurch ausgelesen werden.

Um die Tauglichkeit dieser neuen Architektur zu demonstrieren, ist im Basisdokument die genaue Spezifikation eines Ein-Chip Prototyp-Rechners, der für den Evolutionsbereich gebaut wurde, präzisiert. Pläne für eine Erweiterung dieses Rechners um hundert Chips sind auch im Basisdokument vorhanden. Verschiedene modulare Aufteilungen eines Chips in Module von 2 bis 12 Konfigurationseinheiten sind auch dort aufgezeichnet, um die allgemeine Gültigkeit der Architektur zu zeigen.

2. Vorgang zum Bau eines einem vorbestimmten Problemkreis gewidmeten bzw. an diesen bzgl. seiner Hardware angepaßten Parallelrechners

a) Man identifiziere für ein Problembeispiel die Datenströme, die die einzelnen Vorgänge (Prozesse) koppeln. Diese bilden dann eine bestimmte Anzahl von bitbreiten Ein- und Ausgängen. Man faßt unterschiedliche Problembeispiele zu einem Problemkreis zusammen und untersucht, ob es für diesen Problemkreis eine Obergrenze bzw. Obermenge von zur Lösung der Probleme erforderlichen Prozesse gibt. Wenn nicht, muß man entweder das Verfahren beenden oder nach einfacheren Vorgängen in den Problemen oder nach kleineren Problemkreisen suchen.

b) Man bestimme die Geometrie der Koppelung zwischen den einzelnen Vorgängen, typische Beispiele sind ein-, zwei- oder dreidimensionale Gitter oder der Hyperkubus. Wenn es keine Geometrie gibt, die alle jene spezifischen Problemgeometrien durch Streichen von Verbindungen als Subgeometrien enthalten, dann kann dieses Verfahren einen einzigen Parallelrechner für den Problemkreis nicht liefern.

c) Mit Hilfe eines Blockschaltbildeditors entwerfe man einen Schaltkreis, der die lokale Prozessierung für ein bestimmtes einfacheres Problembeispiel beschreibt. Man lasse dieses Design (mit Hilfe eines automatischen "Place and Route"-Programms) in eine FPGA Teilkonfiguration konvertieren und verpacke das Design in die kleinstmögliche kompakte Teilform aus der Abbildung N des Basisdokuments.

d) Man wähle eine der optimalen Verpackungen dieser Prozessoren aus. Abhängig von der Prozessor- und Chipgröße sind zwischen 16 und 1024 Prozessoren pro Chip möglich. Man zähle die Ein- und Ausgänge, die zwischen den verschiedenen Chips notwendig sind, um diese Prozessorverbindungsstruktur über der Chipgrenze hinaus zu realisieren.

e) Auf "Printed Circuit Boards" verlege man Leitungen, um die Obermenge der in Punkt 2d) bestimmten Verbindungen für die verschiedenen Probleme des Problemkreises zu realisieren. Zusätzlich werden einige Clock- und Steuersignale zum Konfigurieren der Chips gebraucht.

f) Ein DMA Zugriff wird auf einen Datenbus des Hostrechners angeschlossen. Die ausgewählten Eingänge sollten zum Laden des Chips (siehe oben) und die ausgewählten Ausgänge zum Auslesen des verteilten Speichers geeignet sein.

g) Man schreibe Interfacesoftware, um Problemänderungen auf Kommandos des FPGA Designeditors abzubilden. Hier sind zum Beispiel Macros, die

äquivalente Änderungen auf allen Prozessoren hervorrufen, besonders von Nutzen. Die Chipgrenzen können auch transparent gemacht werden, so daß die Verbindungsstruktur leicht in Software zu ändern ist.

Erläuterungen zum Verfahren

FPGAs (Field programmable gate arrays) bilden die Bausteine dieses neuen Designverfahrens. Diese an sich bekannten Chips sind wiederholt konfigurierbar und bestehen u. a. aus einem quadratischen Gitter kombinatorischer Logikblöcke (die mehrere Gatter und Speicherelemente enthalten) und einem verbindenden Netz von programmierbaren Leitungen (Interconnect-Ebene mit programmierbaren Schalter-Matrixen). Das hier vorgestellte Designkonzept macht Gebrauch von einem regelmäßigen Gitter von ca. 100 dieser Chips. Die Aufgabe jedoch, ein funktionierendes Design für nur einen solchen Chip zu entwerfen, birgt normalerweise u. a. komplexe Probleme bei der zeitlichen Abstimmung. Selbst das Äquivalent eines effizienten automatischen Compilers für serielle Probleme ist noch nicht denkbar. Weiterhin fordert das Problem der Kommunikation zwischen Prozessen, die auf den verschiedenen Chips laufen, jedesmal eine gesonderte und schwierige Behandlung.

Die beiden Probleme von Design und Kommunikation werden erfindungsgemäß gelöst durch das hier vorgestellte Verfahren. Die Lösung basiert auf kleinen designenden Schaltkreisen, die einfach durch Hardware-Programmierung zu entwerfen sind und so klein sind wie die kleinsten Prozeßeinheiten der zu lösenden Probleme des Problemkreises. Viele solcher Einheiten werden symmetrisch in jeden Chip verpackt. Kommunikation erfolgt über die Wiederholung eines lokalen Musters von Verbindungslinien zwischen diesen Einheiten. Die Architektur der festen Interchipverdrahtung spiegelt eine Obermenge dieser lokalen Struktur wider, um verschiedene Probleme innerhalb desselben Kreises zuzulassen. Die ganzen Einheiten werden synchron getaktet und so gebildet, daß sie stetig alle Eingaben ihrer Nachbarn in einem lokalen Speicher aufnehmen können. Auf diese Weise werden die sonst schwierigen Kommunikationsprobleme beseitigt.

Die hier angebotene Lösung liegt schon sehr dicht an der Ideallösung, jedes Problem direkt auf Silizium zu verwirklichen, die sich anbieten würde, wenn Multi-Chip-Design einfach wäre. Ein Prototyp eines solchen Rechners ist für Evolutionssimulationen vom Erfinder bereits gebaut worden.

3. Verfahren zur Neukonfigurierung

Die Neukonfigurierung von Parallelrechnern mit der im Punkt 1. erwähnten Architektur ist erfindungsgemäß mit einer klassischen Programmieraufgabe vergleichbar.

a) Editieren
Mit Hilfe einer höheren Computersprache wie z. B. der Sprache C wird die systematische Benennung von Netznamen (der entsprechende Begriff zur Variablen in konventioneller Programmierung) gemacht. Systematische Änderungen, entweder in den Modulen selbst oder in ihrer Verbindungsstruktur, sind dann über programmerzeugten Designmodifikationskommandos, die in Makrodateien zusammengefaßt werden können, effizient durchzuführen.

b) Compilieren
Auf diese Kommandodatei sowie die ursprüngliche Designdatei kann dann von einem Optimisierungs-Routeprogram (vom FPGA Hersteller) zugegriffen werden, um eine neue Designdatei für jedes der FPGAs zu erzeugen. Der Benutzer muß lediglich die lokalen Änderungen in der Verbindungsstruktur oder Logik eines Moduls spezifizieren, um ein neues Gesamtdesign zu erstellen. Aus den Designdateien für die FPGAs werden automatisch anhand von FPGA Hersteller Software ladbare binäre Dateien erzeugt.

c) Laden
Diese Binärdateien werden über die oben in Punkt 3.a) beschriebene Busstruktur parallel oder seriell zu dem Parallelrechner geschickt.

d) Lesen
Die Ergebnisse können entweder direkt aus dem verteilten Speicher gelesen werden (durch einen von FPGA Hersteller initiierten Readbackpulses) oder dynamisch über Datenleitungen, die die Rechnerstruktur an entscheidenden Stellen abtasten.

Nachfolgend wird anhand der Figuren die Erfindung anhand eines Ausführungsbeispiels eines Genetikprozessors näher erläutert.

Kurzbeschreibung der Figuren

Fig. 1 Programmierbare Verbindungseinrichtungen.

Die Darstellung zeigt eine Ecke eines XILINX 4000 LCA-Chip (Logic Cell Array) und die erhältlichen programmierbaren Verbindungen zwischen den CLB (Configurable Logic Block) und den 10-Blöcken (Input/Output-Blöcken), Dekodern und Puffern. Die hohe Dichte der möglichen Verbindungen in den Schaltmatrixen, die ein regelmäßiges Gitter zwischen den CLB bilden, ist besonders auffällig, jedoch zeigen die zahlreichen Punkte auch programmierbare Verbindungspunkte (Programable Interconnection Points — PIP).

Fig. 2 CLB-Konfiguration als Logik oder Speicher.

Hierbei handelt es sich um eine Reproduktion aus dem XILINX Serie XC-4000 Datenbuch. Die Konfigurationslogikblöcke der XILINX Serie 4000 weisen 2 programmierbare Funktionsgeneratoren zur Implementierung einer arbiträren Boole'schen Funktion mit 4 Variablen auf. Diese wird über ein Tabellen-Nachschlagschema implementiert, das nicht nur bei der Konfiguration, sondern auch während des Betriebs programmierbar ist, so daß die Funktionsgeneratoren auch als RAM-Speicher verwendbar sind. Die CLB sind ebenfalls mit bei der Konfiguration programmierbaren Multiplexern versehen, die verschiedene Verbindungen zwischen den direkten und den (durch Flipflops) gehaltenen Ausgängen und den Funktionsgeneratoren ermöglichen. Ein dritter Funktionsgenerator verleiht dem CLB weiteres kombinatorisches Potential oder weiteres Potential als kombinierte RAM. Die Details der schnellen Übertragungslogik, welche die effiziente Verwendung der CLB als Konstruktionsblöcke für Zähler und Addierer oder dergleichen erlauben, sind nicht dargestellt.

Fig. 3 Einheitsstruktur und IO für den Genetik-Prozessor.

Diese Darstellung betrifft die lokale Verbindungsstruktur für 4 • 4 Einheiten des Genetik-Prozessors (wo-

bei 164 CLB dargestellt sind). Die räumliche Isotropie wird bewahrt, indem 4 gedrehte Einheiten (x1, x2, x3, x4) kombiniert werden, um eine übergeordnete Wiederholungseinheit zu bilden, die in den gestrichelten Rechtekken dargestellt ist. Dieses Design paßt gut in den LCA-Chip 4003 der Firma XILINX, während der LCA-Chip 4010 100 = 5 • 5 • 4 Einheiten (400 CLB) aufnehmen kann. Die diagonalen Verbindungen zeigen das Prototyp-Interaktionsschema für die Schaltsteuerung.

Fig. 4 Design der Prototyp-Genetik-Prozessoreinheit.

Die Figur zeigt ein Design einer 4-CLB-Prototyp-Einheit mit 4 Eingängen und 2 Ausgängen. Das Design verwendet einen 2-Ketten-Speicher (FIFO) mit genetischer Steuerung des Informationstransfers von i1, i2 nach o1, o2 über den Schalter SW1. Der Schalter weist vier Stellungen auf und es erfolgt eine interne und externe Steuerung, wenn geschaltet wird. Eine Adressierung wird durch 4 solcher Blöcke aus einem einzelnen globalen zyklischen Adreßzyklus 0000100110101111 erzeugt, wobei ein Flipflop in der Steuerung (CTL) jedes Blocks verwendet wird.

Fig. 5A, 5B, 5C und 5D CLB-Designs für die Genetik-Prozessoreinheit.

Die CLB-Einheit gemäß **Fig. 4** besteht aus den Blökken KAM, MEM, CTL und SWI. Die ersten beiden CLB, nämlich RAM und MEM (**Fig. 5A und 5B**), dienen als zwei Schieberegister und der dritte CLB, nämlich CTL (**Fig. 5C**), steuert die Funktion des vierten CLB, nämlich SWI (**Fig. 5D**), der ein genetischer Schalter für die beiden Bitströme ist.

Fig. 6 LCA-Leitungsführung für den Genetik-Prozessor-Prototyp.

Die Darstellung zeigt die "Verdrahtung" des LCA-Prototyp-Designs zum Zeitpunkt des Einschreibens. Diese Verdrahtung ist mit **Fig. 3** zu vergleichen. Die in **Fig. 5** dargestellten 4 CLB, welche die Grund-Funktionseinheit bilden, sind wie in **Fig. 4** angeordnet. Dieses Verbindungsschema wird automatisch durch den XILINX-Optimierungs-Leitungsführer gemäß einem halbautomatischen Verfahren zur CLB-Einführung und zur Netzbenennung unter Verwendung des grundlegenden 4-CLB-Makros durchgeführt. Die beiden mittleren Reihen und Spalten der CLB werden bei diesem Demonstrations-Design zur Erzeugung der globalen 1-Bit-Adressensequenz, der Bitströme, mit welchen die Einheiten zu Anfang geladen werden, der Logik zum Abtrennen von dieser Ladequelle und einiger anderer, weniger wichtiger Anzeigeverbindungsaufgaben, die für ein endgültiges Design nicht erforderlich sind, verwendet.

Fig. 7 Genetik-Prozessor auf einer XILINX-Demonstrationsplatte.

Die Darstellung zeigt die Pinverbindungen und deren Bedeutung in dem Fall, daß der 4003 LCA-Chip mit dem Design des Genetik-Prozessoreinheit-Prototyps geladen ist. Dip-Schalter auf der Demonstrationsplatte steuern den Abtrenn- und Schalter-Rücksetz-Halteprozeß, während LED und segmentierte Anzeigen zur Angabe der Binärzustände verschiedener repräsentativer Signale verwendet werden, die zwei kollidierende Bitströme und den Zustand des genetischen Schalters an einem Punkt der LCA einschließen.

Fig. 8 Interface zum Host-Speicher als Strömungsreaktor.

Die Figur zeigt ein schematisch dargestelltes Interface, das erreicht wird, indem die Grenz-Bitstromverbindungen zu einem Host-Bus geöffnet werden. Weitere Details finden sich in der nachfolgenden Beschreibung. Es sind ferner die Ladebitstrompfade durch das Gitter aus 4005-Chips dargestellt.

Fig. 9 Einheitsgrößen und Packung bei 14 • 14 CLB LCA (4005).

Es wird bei den Einheitsmustern davon ausgegangen, daß jede LCA ein identisches Format hat. Die (2 • 2)-Wiederholung ist daher erforderlich, um alternierende I/O-Richtungen zu ermöglichen, wodurch Isotropie sichergestellt ist. Bei den komplexesten Designs (3 und 5) müssen 4 Varianten der Grundeinheit geroutet werden.

3:4 • 4 • (2 • 2) • 3 = 64 3-CLB-Einheiten
4:3 • 3 • (2 • 2) • 4 = 36 4-CLB-Einheiten
5:3 • 3 • (2 • 2) • 5 = 36 5-CLB-Einheiten
9:2 • 2 • (2 • 2) • 9 = 16 9-CLB-Einheiten
12:2 • 2 • (2 • 2) • 12 = 16 12-CLB-Einheiten.

Fig. 10 Lokale Adressenerzeugung mit Einfügung und Löschen.

Diese Zwei-CLB-Konfiguration erzeugt eine lokale Adresse für eine separate Schreibpositionierung und ist ferner in der Lage, diese Adresse in einem einzelnen Taktzyklus um zwei zu verzögern oder vorwärts zu bewegen, wodurch Einfügen und Löschen in dem ausgehenden Bitstrom möglich sind.

Die nachfolgende Beschreibung ist wie folgt unterteilt: Im Abschnitt 1 wird die Beschaffenheit einer Familie von gegenwärtig erhältlichen FPGA dargestellt, und zwar die Serie 4000 der Logic-Cell-Arrays (LCA) der Firma XILINX. Im Abschnitt 2 wird das verwendete Prototyp-Design beschrieben. Im Abschnitt 3 wird die Entwicklung einer ausreichend flexiblen Parallel-Programmierumgebung zur Modellerstellung mit einer festen Zahl von LCA erörtert. Im Abschnitt 4 wird ein generischer genetischer Prozessor, der diese Methodik verwendet, vorgestellt. Schließlich wird im Abschnitt 5 die Art der Simulationsexperimente, die mit dieser Art von Computer durchführbar sind, sowie die Möglichkeit des Restrukturierens erörtert, um einen Kompromiß zwischen Populationsgröße und Rechnergeschwindigkeit zu erhalten.

1. Die XILINX LCA 4000 Serie

Die Logikzellenanordnungen von XILINX sind auf CMOS VLSI Technologie basierende anwenderprogrammierbare Logikfelder. Der Ausdruck "programmierbar" bezieht sich auf die Tatsache, daß die Schaltung geringster Ordnung auf multipotente Weise konstruiert ist, die durch das Konfigurieren zahlreicher interner Speicherzellen (z. B. 178096 bei dem größten gegenwärtig erhältlichen LCA, dem XC-4010) näher spezifizierbar ist. Diese spezifizieren sowohl eine Vielzahl logischer Elemente als auch deren Verbindungen: primär die Anordnungen der konfigurierbaren Logikblökke (CLB) und der Schaltmatrixen. Darüber hinaus ermöglichen Input-Output-Blöcke (IOB) und breite Detektoren eine programmierbare Kommunikation mit den externen Pins des Chips, und programmierbare Verbindungspunkte (PIP) und Dreizustands-Puffer (TBUF) spezifizieren die Art der Verbindung zwischen den genannten Elementen, wobei eine Vielzahl von zusätzlichen lokalen und globalen Verbindungsleitungen (Leitungen mit einfacher Länge, Leitungen mit doppelter Länge, horizontale und vertikale Langleitungen und globale Leitungen) verwendet wird. In den Ecken des Chips befinden sich eine Anzahl zusätzlicher spezieller

Strukturen, von denen insbesondere einige zur Steuerung der seriellen Programmierung, des Starts und der Wiederholfunktion des Chips verwendbar sind. Die in einem Teil der LCA verfügbaren Mittel sind in Fig. 1 dargestellt. Ein Seriell-Konfigurationsprogramm (das, wie in Abschnitt 4 dargelegt, mittels eines dem Kompilieren verwandten Vorgangs erzeugt wird) kann aus einem herkömmlichen PROM-(programmierbaren Lesespeicher)-Chip oder von einer Platte über einen Standard-PC oder eine Workstation geladen werden.

Die CLB können entweder als logische Elemente oder als KAM (Direktzugriffsspeicher) oder als Kombination aus beiden programmiert sein, wie in Fig. 2 gezeigt. Als logisches Element enthält der CLB zwei unabhängige Funktionsgeneratoren (F und G) mit 4 Eingängen und einen dritten Funktionsgenerator (H) mit drei Eingängen, die Ausgänge von F und G und einen neunten externen Eingang. Diese drei Funktionsgeneratoren sind derart programmierbar, daß jegliche Boolesche Funktion ihrer Eingänge realisierbar ist. Jeder dieser drei Ausgänge oder ein zehnter Eingang ist mit den beiden direkten Ausgängen des CLB verbindbar oder durch einen der beiden Flipflops (Einzelbitspeicherung) in dem CLB leitbar, die als weitere externe Eingänge des CLB mit Takt-, Setz-/Rücksetz- und Freigabe-Steuerungen verbunden sein können. Darüber hinaus ist der CLB mit programmierbarer schnellverarbeitender Logik versehen, die die Verwendung schneller Zähler und dergleichen über mehrere CLB ermöglicht. Als Speicherelement kann der H-Funktionsgenerator verwendet werden, um die Verwendung von F und G entweder als separate 16-Bit-Speicher (wobei die 4 Eingänge als Adreßleitungen agieren) oder als kombinierter 32-Bit-KAM zu steuern. (Als zwei separate Speicher wird der Schreibzugriff für beide jedoch von einem einzelnen Schreib-Freigabesignal gesteuert.) Die ausgelesenen Bits können wie zuvor dargelegt sowohl direkt als auch durch die Flipflops ausgegeben werden. Dieses Merkmal ist ein wesentlicher Fortschritt der Serie 4000, da es das Potential einer programmierbaren dichten parallelen Datenverarbeitung unter Verwendung verteilter Speicher ermöglicht. Die Wiederholungsfunktion liefert eine Seriell-Bitsequenz mit den gegenwärtigen Zuständen sämtlicher Flipflops, einschließlich der in den IOB nicht erwähnten. Der Inhalt wird bei der Initiierung der Wiederholfunktion kopiert, so daß eine normale LCA-Verarbeitung parallel zu der Wiederholfunktion ablaufen kann. Jedoch werden die Inhalte der als KAM ausgelegten CLB nicht kopiert, so daß der Benutzer die Schreib-Freigabe für alle KAM, deren Werte bei der Wiederholfunktion benötigt werden, synchron herunterziehen muß.

Die Leitungsführung der Benutzerverbindungen, die Verwendung der Schaltmatrixen und der genannten Verbindungseinrichtungen, erfolgt zu einem großen Teil automatisch mit einem Optimierungscode, der ein simuliertes Ausheilen beinhaltet, jedoch muß die Verteilung bestimmter hoher Fan-Out-Signale, einschließlich der Taktsignale und der zuvor erwähnten Schreib-Freigabe, auf die Verwendung globaler Leitungen beschränkt werden, um mit übermäßiger Belastung einhergehende größere Zeitverzögerungen zu vermeiden. Diese Leitungsführung kann ebenfalls als eine compiler-ähnliche Funktion angesehen werden. Es sei darauf hingewiesen, daß der eine LCA-Chip durch den Benutzer mittels einer Seriell-Bitsequenz reprogrammierbar ist, um zahlreiche verschiedene Logikschaltungen zu implementieren. Die Bitsequenz ist das Ergebnis der Kompilierung einer Design-Datei, die ihrerseits das Ergebnis des Routens der von dem Benutzer eingegebenen Spezifikationen bezüglich Logik und Verbindungen ist. Es sind keine zusätzlichen Modifikationen der Hardware erforderlich, vorausgesetzt, das Design stellt keine neuen Anforderungen an externe Einrichtungen, wie zum Beispiel Drahtverbindungen zu den IO-Pins des Chips. Weitere Einzelheiten finden sich in den XILINX Teile-Spezifikationsangaben.

2. Design-Prototyp für die Genetik-Datenverarbeitung

Genetische molekulare Sequenzen werden modellhaft als binäre Ketten variabler Länge dargestellt. Zunächst sei darauf hingewiesen, daß die Subsequenz 00 als Separator zur Markierung der Lücken zwischen den Ketten verwendet wird. Die Ketten sind in Schieberegister eingebettet, die in einem regelmäßigen Gitter in der gesamten LCA eingebettet und paarweise an einer Knotenpunktanordnung verbunden, die als genetische Schalter bezeichnet werden. Der Datenfluß der Ketten durch die Schieberegister und die Schalter ist konstant die Taktgeschwindigkeit (die bis zu $3 \cdot 10^7$ Bits/sek beträgt). Die Schalter steuern einfach die Art und Weise, in der von zwei konvergierenden Schieberegistern eingehende Bits auf die beiden divergierenden Schieberegister verteilt werden. Das Muster des Datenflusses im vorgegebenen Schalterzustand ist in Fig. 3 dargestellt. Dieses wird gewählt, um die Schieberegister anfänglich zu laden und um räumliche Symmetrie während des freien Schaltens zu bewahren. Bei der einfachen Prototypversion erreicht entweder einer der beiden oder beide eingehende Bitströme die beiden ausgehenden Ströme (Kopie oder Auslagerung), und der Schalter steuert ferner, welcher der beiden Eingangsströme die Ausgänge erreicht. Somit weist der Schalter 4 Zustände auf, die von zwei Flipflops gesteuert werden. Veränderungen im Zustand des Schalters treten nur auf, wenn die Schalter-Flipflop-Taktgebung freigegeben wird. Bei dem Prototyp-Design ist die Freigabe aktiv, wenn beide Eingangsströme eine Lücke zwischen den Ketten (00 Subsequenz) aufweisen, und der neue Schalterzustand wird sodann durch zwei Bits eines dritten benachbarten Bitstroms bestimmt (der Dateneingang der Flipflops). Wenn eine Übereinstimmung von 3 Bits zwischen den gegenwärtigen Bits in einem Eingangsstrom und denjenigen des dritten Nachbarn gegeben ist, wird der Schalter ebenfalls freigegeben. Letzterer ermöglicht das Schalten innerhalb von Ketten und somit eine breite Palette von rekombinatorischen und mutativen Ereignissen, einschließlich des Teilens und des Verbindens. Es sei darauf hingewiesen, daß bei dem Prototyp-Design das Einfügen oder Löschen einzelner Schritte in das bzw. aus dem Inneren der Ketten nicht zulässig ist. Der Vorteil dieses Designs ist, daß es einen einheitlichen einfachen Rahmen zum Mischen, Verstärken und lokalen Selektieren schafft.

Eine quadratische lokale Einheit von 4 CLB genügt zum Implementieren zweier 34-Bit Zwischenschalter-Kettensegmente, dem genetischen Schalter und dessen Steuerung. Das Block-Design dieser Einheit ist in Fig. 4 dargestellt. Zwei in den Fig. 5A und 5B dargestellte CLB (KAM und MEM) implementieren zwei benachbarte $2 \cdot 17$-Bit- Schieberegister, welche die F- und G-Funktionsgeneratoren als Speicher und die Flipflops zum Speichern des 17ten Bits verwenden. Das Adressieren der 16-Bit-Speicher erfolgt in einem Zyklus mit 16 Taktschritten, wobei pro Taktschritt ein Bit ausgelesen und

geschrieben wird. Normalerweise würde für diesen Zweck ein 4-Bit-Zähler verwendet, jedoch ist ein Zyklus mit 16 verschiedenen 4-Bit-Adressen auch durch konsekutives Überlappen von 4-Bit-Subsequenzen eines einzelnen Bitstroms erzielbar und dies verringert die Zahl der Adreßsignale, die global durch den Chip zu verteilen sind, von 4 auf 1. Die verwendete Sequenz lautete 0000100110101111 und wurde über einen ersten globalen Puffer von der Ecke des Chips zu jeder Gruppe von 4 lokalen Einheiten hin verteilt, wobei ein Flipflop aus jeder der 4 Einheiten zum Wiederherstellen der gegenwärtigen 4-Bit-Adresse verwendet wurde. Ist dieses Prinzip einmal verstanden, so sind ähnliche Adressiervarianten leicht zu entwickeln. Der einzige Unterschied zwischen den 4 lokalen Einheiten, welche die in Fig. 3 dargestellte eigentliche LCA-Wiederholungseinheit (mit 8 Ketten und 4 Schaltern) bilden, liegt in der Orientierung der Eingänge und der Ausgänge.

Der Schalt-CLB (SWI) ist in Fig. 5D dargestellt. Die beiden Flipflops weisen eine Rücksetzleitung auf, die zum Halten des Schalters in dessen Vorgabeposition während des Initialisierens dienen, sowie eine Schreibfreigabe, die, zusätzlich zu den von den oben bezeichneten dritten benachbarten Bitströmen her kommenden Dateneingängen, von dem Steuer-CLB (CTL) her kommen. Einer der Ströme muß den H-Funktionsgenerator durchlaufen, um sein Flipflop zu erreichen. Die Flipflops spezifizieren die 4 möglichen Zustände des Schalters:

1. SWAP1: 00 i1—o1 und i2—o2
2. SWAP2: 01 i1—o2 und i2—o1
3. COPY1: 10 i1—o1 und i1—o2
4. COPY2: 11 i2—o1 und i2—o2.

Der Rest des CLB (die F- und H-Generatoren) werden zum Implementieren des Schalters verwendet, wobei jeder Funktionsgenerator einen Ausgang spezifiziert. Die vier Eingänge der Funktionsgeneratoren sind in beiden Fällen gleich: die beiden Eingangsbitströme und die beiden Schalterzustandsausgänge der Flipflops.

Schließlich verwendet der Steuer-CLB (CTL) von Fig. 5C einen Funktionsgenerator zum Untersuchen auf eine doppelte Lücke in den beiden eingehenden Bitströmen (00 und 01) und den anderen zum Untersuchen auf eine Zwei-Bit-Übereinstimmung zwischen dem ersten eingehenden Strom und dem benachbarten dritten Strom. Ein Flipflop wird zum Aufzeichnen des vorherigen Zustands dieser letztgenannten Übereinstimmung verwendet und sodann zusammen mit den Ausgängen von F und G zur endgültigen Entscheidung in der Logik von H zurückgeleitet (2 aufeinanderfolgende Übereinstimmungen (d. h., eine 3-Bit-Übereinstimmung) zwischen dem ersten und dem dritten Bitstrom oder eine doppelte Lücke in Strom 1 und 2), so daß das Schreib-Freigabesignal eine Veränderung des Schalterzustands ermöglichen kann. Das verbleibende Flipflop wird zur vorgenannten Adressendekodierung verwendet.

Das Prototyp-Design wurde auf dem kleinsten Chip (4003) der Serie 4000 implementiert, für den eine Demonstrationsplatine von XILINX zur Verfügung stand. Diese weist eine 10 • 10 Anordnung von CLB auf und ermöglicht die Verwendung von 64 CLB für 16 zu 4 Gruppen verbundenen lokale Einheiten (32 34-Bit-Ketten und 16 Schalter), wobei 2 CLB-breite vertikale und horizontale Streifen in der Mitte zum Testen und für einfache On-Chip-Erzeugung der globalen Steuerungen frei blieben (eine Off-Chip-Lösung wird für größere Designs vorgeschlagen). Die Leitungsführung der LCA ist in Fig. 6 dargestellt. Nur eine kleine Untergruppe der 84 Pins ist zur Vervollständigung einer toroidalen Geometrie von Bitströmen zum Testen des Einzelchips erforderlich. Die Details der Verbindungen und der Steuerschalter, die bei diesem Demonstrations-Design verwendet wurden, sind in Fig. 8 dargestellt.

Für das endgültige Design sind 100 der großen Chips (20 • 20) erforderlich, die auf einer Leiterplatte (PCB) verdrahtet sind, um eine Population von 10000 Ketten mit einer durchschnittlichen Länge von 64 Bits zu erreichen.

Die begrenzte Populationsgröße dieses Genetik-Prozessors kann teilweise dadurch behoben werden, daß ein Öffnen der geschlossenen Prozessor-Bitströme zum Haupt-KAM-Speicher des Host-Computers mit variabler Bitbreite erfolgt. In zwei Host-Buszyklen können bis zu 32 Bits (je nach Host-Bus) parallel in die LCA-Anordnung ein- und ausgegeben werden. Eine große Population von 1 Million 64-Bit-Ketten (8MB) oder mehr kann dann verarbeitet werden, indem die Speicheradresse linear durch den Host-Speicher getaktet wird, wobei die in die LCA eingegebenen Bits durch die von dieser ausgegebenen ersetzt werden. Je nach der Verweilzeit der genetischen Ketten in der LCA sind zahlreiche effektive Populationsgrößen erzielbar, indem die Bitbreite des Interface variiert wird. Eine schematische Darstellung dieses Host-Interface ist in Fig. 8 dargestellt.

3. Programmierumgebung für Hardware-Simulation

Üblicherweise liegen zwischen dem Design einer Schaltung und deren Implementierung Zeiträume in der Größenordnung von Monaten. Die optimierte Schaltung muß einer Chip-Herstellerfirma übergeben werden, die sie mittels einer Abfolge von photolithographischen und chemischen Schritten mit einer Auflösung von 1 Mikrometer auf ausgesuchte Silizium-Wafer aufbringt. Die Kosten eines solchen Entwicklungszyklus sind hoch und nur durch große Verkaufsmengen identischer Chips abzudecken. Dies stellt das Entwicklungshindernis für kundenspezifische Chips dar. Bei den FPGA hat ein bestimmtes regelmäßiges und dichtes Design den genannten Entwicklungszyklus durchlaufen und seine Schaltung weist Speicherzellen (Flipflops) auf, deren binäre Zustände über Transistoren das Öffnen und Schließen einer großen Zahl von Verbindungen zwischen Design-Komponenten (den PIP) und auch kombinatorische Funktionen ermöglichen, welche eine Gruppe von Signalen und eine Ausgangsleitung verbinden (z. B., die F-, G- und H-Funktionen der Serie 4000 von XILINX). Daraus ergibt sich, daß die Spezifikation der Zustände dieser Speicherzellen den gleichen Effekt hat, wie das Ausführen einer sehr großen Klasse von Schaltungsdesigns in Hardware, mit einer Verringerung der erreichbaren Gatterdichte um etwa den Faktor 10. Das Schaltungsdesign eines Chips der Serie 4000 wird durch mehrere 10 000 Speicherzellen gesteuert.

Selbstverständlich können damit nicht alle Verbindungen spezifiziert werden. Die FPGA weisen eine regelmäßige Anordnung dichter lokaler Verbindungsmöglichkeiten und eine begrenzte Anzahl schneller Verbindungsleitungen größerer Reichweite auf. Es wurde besonderes Augenmerk auf die Auswirkungen der programmierbaren Verbindungen auf die Ausbreitungszeit und den Spannungspegel der Binär- und der Dreizustandssignale gelegt, jedoch lassen sich diese Probleme auf dieser Stufe nicht von dem Benutzer trennen. Andererseits ist es sehr einfach, Programme in höheren Computersprachen zu erzeugen, die ebenfalls nicht funktio-

nieren. Es sind daher ein intelligentes Design-Eingangsinterface und ein intelligenter Compiler erforderlich, die Fehler in einem Design begrenzen und den Nutzen eines Designs testen. Da das Produkt mit herkömmlichen Chip-Designs konkurriert, wurde großer Wert auf einen Design-Eingang mit herkömmlichen CAE-Block-Designverfahren gelegt. Das Block-Design ist jedoch völlig allgemein und muß später an die Programmiereinschränkungen der LCA angepaßt werden. Der erfahrene Benutzer kann Beschränkungen in diesem Prozeß spezifizieren, jedoch hängen die Packungsdichte und die Geschwindigkeit (die maximale erzielbare Taktrate) des Designs oftmals von globalen Gesichtspunkten ab, die den derzeit erhältlichen automatischen Implementierungsprogrammen, z. B. dem von XILINX angebotenen PPR (Partion place and route) entgehen.

In Übereinstimmung mit dem Konzept der FPGA als regelmäßige Logikanordnungen erfolgte eine Beschränkung auf Designs, die eine regelmäßige Wiederholungseinheit beinhalten. Aufgrund der Symmetrie des Chips ist das Problem der Optimierung auf die Zelleinheit dieser wiederholenden Designstruktur beschränkt. Eine Hierarchie verschachtelter Designs, die verschiedene kleinere Einheiten zu einer größeren Wiederholungseinheit kombiniert, kann ebenfalls in Betracht gezogen werden. Die Software von XILINX unterstützt bisher noch nicht anwenderdefinierte "harte" Makros, die das Aufbringen vollständig optimierter und bestimmter Subdesigns auf einem Chip ermöglichen, jedoch wird dies als imminent vorausgesetzt. Bei regelmäßigen Anordnungen von Subdesigns ist eine systematische Bezeichnung von Netzen (Signalleitungen) im Subdesign erforderlich, um Duplikationen von Bezeichnungen in internen Signalen zu vermeiden und eine systematische programmierte Verbindung von Subdesign-Eingängen und -Ausgängen zu ermöglichen.

Es wurden in C geschriebene Programme entwickelt, um das systematische Set von Netzverbindungsbefehlen zu schaffen, die zum Implementieren einer gewünschten regelmäßigen Topologie, wie der des Prototyp-Designs, erforderlich sind. Es wurde ebenfalls ein in C geschriebenes Programm entwickelt, um die erforderliche Sequenz von "weichen" Makrobefehlen zu erzeugen, die zur Anordnung der Subdesigns auf der LCA erforderlich sind. Da der XILINX LCA Design-Editor (XACT) das Aufrufen von Befehlsdateien ermöglicht, kann der Schritt von den kleinen Subdesigns zu regelmäßigen Anordnungen solcher Subdesigns automatisiert werden, wenn solche in C programmierten Befehlsdateien verwendet werden. Ein besonderes Verbindungsschema (Geometrie) kann sodann in der herkömmlichen Simulationssprache spezifiziert und automatisch auf den Chip übertragen werden. Das Problem einer weitergehend automatischen Optimierung und eines solchen Testens der Subdesigns sei hier vernachlässigt, jedoch sei gesagt, daß ein Maximum von 16—25 CLB als die primitive Einheit für diese Klasse von Computern angesehen wird. XILINX liefert sehr ausgereifte Validierungskontrollen beim Einführen von Designs, wenn gewünscht, mit inkrementaler Leitungsführung sowie eine Designregel-Überprüfeinrichtung (Design Rules Checker — DRC), die weitere Fehler auffindet. Es existiert eine automatische Routine, welche die maximalen Verzögerungen für die Signalausbreitung zwischen Flipflops angibt, wodurch eine Bestimmung der maximalen Taktgeschwindigkeit und weitere Design-Optimierungen möglich sind. Eine Zeitgebungs-Simulation des Designs kann, wenn gewünscht, ebenfalls in Software ausgeführt werden, wobei die herstellungsbedingten Verzögerungsspezifikationen zusätzlich zur Berechnung des Signalausbreitungsverhaltens verwendet werden.

Die Ausbreitung sich wiederholender Einheiten über die Chipgrenzen hinaus erfordert das Einbringen der geometrischen Verteilung der Pins in das C-Programm, welches die logischen Verbindungen zwischen den Subdesigns an der Peripherie zu den IO-Pins des Chips herstellt. Dies ist eine "Once-off"-Aufgabe für jedes Mitglied der 4000-Familie. Die physikalischen Verbindungen zwischen den Chips müssen durch eine PCB erfolgen, d. h., durch harte Verdrahtung; da jedoch die internen Verbindungen den Subdesign-IO und den IO-Pins der Vorrichtung programmierbar sind, genügt eine große Zahl paralleler Verbindungen zwischen entsprechenden IO-Pins, um den allgemeinen Charakter der Wiederholung der Designs von Einheiten über mehrere Chips aufrechtzuerhalten. Da nur benachbarte Chips miteinander verbunden sind und eine minimale Auffächerung gegeben ist, sollten Verzögerungen kurz genug sein, um die Taktgeschwindigkeit des Computers nicht zu beschränken. Es wurde ein C-Kode geschrieben, um den einer vorbestimmten angeordneten Subdesign-IO-Leitung nächsten Verbindungspin aufzufinden und die entsprechenden XACT-Verbindungsbefehle zu erzeugen.

4. Ein allgemeiner genetischer Schalter

In den vorangehenden Abschnitten wurde ein großer Teil der allgemeinen Design-Strategie dargelegt. In diesem Abschnitt soll der allgemeine Charakter des Ansatzes verdeutlicht werden, indem das Design einer genetischen Schalteinheit mit separater Schreibadressierung (einem Schreibkopf) dargelegt wird, welche in Abhängigkeit von lokalen Bit-Sequenzcharakteristika (einer der beiden durch die Einheit laufenden Bitströme oder auch ein benachbarter Strom) schrittweise in bezug auf die Leseadresse bewegbar ist. Eine Anordnung solcher Elemente, die Turing-Maschinen ähnlich sind, erlaubt eine generelle Bearbeitung der eingebetteten genetischen Sequenzen, einschließlich des Einfügens und Löschens innerhalb derselben. Fig. 9 zeigt, wie Einheiten in der Größe von 3, 4, 5, 9 und 12 CLB gleichmäßig in der 14 • 14 4005 LCA eingebettet werden können. Andere Einheitsgrößen sind mit der 4005 LCA möglich, wenn die Beschränkung bezüglich einer geraden Zahl von Einheiten pro Seite gelockert wird (dies würde jedoch in der endgültigen Anordnung alternierende Varianten des Designs für benachbarte LCA erfordern, um räumliche Isotropie sicherzustellen). In diesem Abschnitt wird eine Einheit mit 9 CLB dargelegt.

Es werden hierbei große Teile der Prototypstruktur beibehalten, welche die genannten 4 CLB betreffen (KAM, MEM, CTL und SWI), und die neue Struktur wird als Erweiterung eingebracht. Die Schreibadresse muß lokal gespeichert sein (wobei 4 Flipflops und 1 Funktionsgenerator in 2 CLB ausreichen), um die hier gewählte 4-Bit-Adresse zu erzeugen. Es wird die zuvor dargestellte normale rotierende Bitsequenz von Adressen verwendet. Die Adresse zum nächsten Taktzeitpunkt kann gegenüber der erwarteten nächsten Adresse inkrementiert oder dekrementiert werden, indem die Zahl der Taktimpulse an den Flipflops von 1 zu 2 oder 0 verändert wird. Die einfachste Art der Vorbewegung der Adresse um einen Schritt über die nächste Adresse hinaus ist die Verwendung eines Doppelfrequenztakts zur Erzeugung zweier Zyklen in diesem Teil der Schal-

tung. Das Freigabesignal an den Flipflops kann auf einem niedrigen Pegel gehalten werden, um eine Aufdatierung zu verhindern und somit eine relative Dekrementierung der Adresse von dem erwarteten nächsten Wert aus zu bewirken. Jedoch ist es möglich, diese zusätzliche Funktion in die beiden CLB zu packen, die zur Erzeugung der lokalen Adresse ohne Verwendung eines Doppelfrequenztakts verwendet werden (welcher zu einer Beschränkung der maximalen Operationsfrequenz führen könnte). Dies ist in Fig. 10 dargestellt. Die Adreßbits 0 und 2 sind in einen CLB gepackt und die Bits 1 und 3 in den anderen. Die schematische Darstellung zeigt die Dichte, mit der sehr komplexe Funktionalität ausgedrückt werden kann. Eine Zwei-Bit-Zerlegung des relativen Adreß-Schiebesignals (das erste Bit für ein Dekrement oder kein Dekrement, das zweite für ein Inkrement oder kein Inkrement, wobei das Dekrementieren Vorrang hat, da es mit der Flipflop-Freigabe verbunden ist).

Das separate Lese- und Schreib-Adressieren eines Bitstroms während dessen Durchlauf durch das zweite 16-Bit-Register (in MEM) erfordert eine Doppeltaktfrequenz, um den getrennten Lese- und Schreib-Zugriff zu ermöglichen. Ein 8 : 4-Bit-MUX ist zur alternierenden Auswahl zwischen den Lese- und Schreib-Adressen erforderlich. Es ist für den Fachmann leicht ersichtlich, wie dieser MUX in 2 CLB vorzusehen ist, so daß hier nicht näher darauf eingegangen wird.

Ein endgültiger CLB kann verwendet werden, um die Abhängigkeit des Inkrementifer-Stationär-Dekrementier-Signals von den lokalen und den benachbarten Bitströmen zu erzeugen, wie dies bei der Steuereinheit (CTL) des Prototyps des genetischen Schalters geschehen ist. Wahlweise ist ein einzelner Funktionsgenerator (H) mit drei Eingängen und ein Flipflop ausreichend, um den Halbfrequenztaktimpuls zu liefern, wenn dies zur Verbesserung der Stabilität erforderlich sein sollte. Diese könnten in einen der beiden MUX-CLB gepackt werden.

Das endgültige Design der Einheit weist somit 9 CLB auf (KAM, geteilter R/W MEM, 2 für die Schreib-Adresse, 2 für den MUX, CTL, SWI und die Inkrementier/Dekrementier-Steuerung). Dies zeigt, wie eine erhebliche neue Funktionalität des Einfügens und Löschens von bis zu 16 Bit mit einbezogen werden kann, während gleichzeitig relativ kleine Einheiten beibehalten werden. Die grundlegende Idee des geteilten Adressierens kann zur weiteren Erweiterung der Funktionalität verwendet werden.

5. Experimentelle Vorschläge für den Genetik-Prozessor

Bei der sehr großen Zahl von zu berücksichtigenden Generationen ist der Computer auf ideale Weise geeignet, Fragen der evolutiven Stagnation und Punktuation in Zeitmaßstäben anzugehen, die mit der Evolutionsgeschichte auf der Erde vergleichbar sind. Insbesondere der schwierige Übergang von unabhängigen Replikatoren zu komplexen funktionell integrierten Komponenten höherer Organismen kann damit untersucht werden (zugegebenermaßen in der hier modellhaft konstruierten abstrakten logischen genetischen Welt). Darüber hinaus können Fragen der Langzeit-Optimierung in interagierenden Systemen untersucht werden. Zunächst müssen die Mischeigenschaften der sequenzabhängigen Auslagerung in dem Prototyp-Modell und sodann die Populationsdynamik der rein replikativen Kopierfunktionalität untersucht werden, bevor die Variation und die Selektion im gesamten Prototyp untersucht werden. Bereits hier ergeben sich interessante Fragen bezüglich der Zeitskala für die Optimierung und die evolutionäre Stabilität.

Nachdem auf diese Weise das Interesse an solchen auf die Ebene von Giga-Generationen erweiterten logischen Evolutionsprozessen begründet ist, können nunmehr andere biologisch relevante und komplexere Interaktionsmuster untersucht werden. Andere Untersuchungen der zahlreichen alternativen genetischen Interaktionsschemata, die zur Untersuchung funktionaler Interaktionen in der Evolution verwendbar sind, können sich anschließen. Der in dem vorhergehenden Abschnitt dargestellte, einem Turing-Schalter ähnliche genetische Schalter gibt eine Vorstellung von dem Komplexitätsniveau, das in kleinen Subdesigns erzielbar ist.

Patentansprüche

1. Vorrichtung zur Berechnung von feinkörnigen parallelisierbaren Problemen, die als eine Vielzahl miteinander verkoppelter diskretisierbarer Prozesse beschreibbar sind, mit

— mehreren programmierbare in Arrayform angeordneten benachbarten Logikschaltungen, die jeweils aufweisen

— mehrere konfigurierbare Logikschaltungsblöcke, die aus regelmäßig angeordneten Logikzellen zusammengesetzt sind und frei konfigurierbar sind,

— mehrere programmierbare Schaltermatrizen zum regelmäßigen Verbinden der Logikschaltungsblöcke untereinander entsprechend der Schaltermatrix-Programmierung,

— mehrere programmierbare I/O-Blöcke mit I/O-Anschlüssen zum regelmäßigen Verbinden der mehreren programmierbaren Logikschaltungen miteinander und zum Verbinden mindestens einer der Logikschaltungen mit einem Host-Rechner,

— wobei mehrere Logikschaltungsblöcke jeweils zu identischen Gruppen zusammengefaßt sind, von denen jeder eine bestimmte im vornhinein vorgegebene, von der Größe der Gruppe von Logikschaltungsblöcken abhängige Anzahl von I/O-Anschlüssen zugeordnet sind, die zum regelmäßigen Verbinden von Gruppen von Logikschaltungsblöcken untereinander sowie zur Verbindung zweier Gruppen von Logikschaltungsblöcken benachbarter Logikschaltungen zur Verfügung stehen,

— wobei die Geometrie der Gruppen zusammengefaßter Logikschaltungsblöcke von Gruppe zu Gruppe gleich ist,

— wobei die Verbindung zwischen den Gruppen von Logikschaltungsblöcken und die Verbindung zwischen den Logikschaltungsblöcken der Gruppen jeweils gleich ist,

— wobei die Verbindung zwischen den einzelnen Logikschaltungen derart ist, daß die Anzahl von Verbindungen, die dem Benutzer aufgrund der Art der Gruppierung von Logikschaltungsblöcken vorgegeben ist, maximal ist,

und die Verbindung von Gruppen von Logikschaltungsblock-Gruppen benachbarter Logikschaltungen gleich der Verbindung der Gruppen von Logikschaltungsblöcken ein und derselben Logikschaltung ist, und

— einer Taktgeneratorvorrichtung, die derart mit den Logikschaltungen und deren Gruppen von Logikschaltungsblöcken sowie mit deren Logikzellen verbunden ist, daß die Taktung sämtlicher Logikzellen global und synchron erfolgt.

2. Vorrichtung zur Berechnung von feinkörnigen parallelisierbaren Problemen, die als eine Vielzahl miteinander verkoppelter diskretisierbarer Prozesse beschreibbar sind, mit

— einer programmierbaren Logikschaltung, die aufweist

— mehrere konfigurierbare Logikschaltungsblöcke, die aus regelmäßig angeordneten Logikzellen zusammengesetzt sind und frei konfigurierbar sind,

— mehrere programmierbare Schaltermatrizen zum regelmäßigen Verbinden der Logikschaltungsblöcke untereinander entsprechend der Schaltermatrix-Programmierung,

— mindestens einen programmierbaren I/O-Block mit I/O-Anschlüssen zum Verbinden der Logikschaltung mit einem Host-Rechner,

— wobei mehrere Logikschaltungsblöcke jeweils zu identischen Gruppen zusammengefaßt sind, von denen jeder eine bestimmte im vornhinein vorgegebene, von der Größe der Gruppe von Logikschaltungsblöcken abhängige Anzahl von I/O-Anschlüssen zugeordnet sind, die zum regelmäßigen Verbinden von Gruppen von Logikschaltungsblöcken untereinander zur Verfügung stehen,

— wobei die Geometrie der Gruppen zusammengefaßter Logikschaltungsblöcke von Gruppe zu Gruppe gleich ist,

— wobei die Verbindung zwischen den Gruppen von Logikschaltungsblöcken und die Verbindung zwischen den Logikschaltungsblöcken der Gruppen jeweils gleich ist,

— wobei die Verbindung zwischen den einzelnen Logikschaltungen derart ist, daß die Anzahl von Verbindungen, die dem Benutzer aufgrund der Art der Gruppierung von Logikschaltungsblöcken vorgegeben ist, maximal ist, und

— einer Taktgeneratorvorrichtung, die derart mit den Gruppen von Logikschaltungsblöcken sowie mit deren Logikzellen verbunden ist, daß die Taktung sämtlicher Logikzellen global und synchron erfolgt.

3. Verfahren zur Konfiguration einer Vorrichtung zur Berechnung von feinkörnigen parallelisierbaren Problemen, die als eine Vielzahl miteinander verkoppelter diskretisierbarer Prozesse beschreibbar sind, bei dem

— ein vorgegebener Kreis von zu berechnenden Problemen als eine bestimmte geometrische Anordnung von gleichen Gruppen von Prozessen dargestellt wird, wobei die Prozesse eines Problemkreises gleich oder unterschiedlich sind,

— die Prozesse und die Verknüpfung der Prozesse des Problemkreises identifiziert und definiert werden,

— einzelne Prozesse zu Gruppen zusammengefaßt und diese Gruppen schaltungstechnisch durch Zusammenfassung und Verbinden mehrerer programmierbarer Logikzellen zu Logikschaltungsblöcken einer oder mehrerer Logikschaltungen realisiert werden und

— die einzelnen identischen Gruppen von Logikschaltungsblöcken der Verknüpfung der Prozesse des Problemkreises entsprechend verbunden werden, wobei diese Verbindungen von Gruppe zu Gruppe und innerhalb der Gruppen von Logikschaltungsblock zu Logikschaltungsblock gleich sind.

4. Verfahren nach Anspruch 3, dadurch gekennzeichnet, daß mehrere programmierbare Logikschaltungen regelmäßig benachbart zueinander angeordnet vorgesehen sind, wobei das Verbindungsmuster zwischen Logikschaltungsblock-Gruppen benachbarter Logikschaltungen gleich dem Verbindungsmuster der Logikschaltungsblock-Gruppen ein und derselben Logikschaltung ist.

5. Verfahren nach Anspruch 3 oder 4, dadurch gekennzeichnet, daß eine Verbindung mindestens einer Logikschaltung zu einem Host-Rechner aufgebaut wird.

6. Verwendung von konfigurierbaren Logikschaltungs-Arrays mit verteilten Speichern zur Simulation naturwissenschaftlicher, betriebs- und volkswirtschaftlicher und logistischer parallelisierbarer feinkörniger Probleme, insbesondere zur Simulation biotechnologischer Systeme, evolvierender Systeme.

7. Verwendung von konfigurierbaren Logikschaltungs-Arrays mit verteilten Speichern zur Berechnung von Problemen der kombinatorischen Optimierung insbesondere des optimalen Routens.

8. Verwendung von konfigurierbaren Logikschaltungs-Arrays mit verteilten Speichern zur Implementierung von genetischen Algorithmen, neuronalen Netzwerken oder anderen verteilten Systemen.

9. Verwendung von konfigurierbaren Logikschaltungs-Arrays mit verteilten Speichern zur Lösung von Problemen des parallel erfolgenden Vergleichens, Veränderns, Sortierens und/oder Aufsuchen von Datenstrings in Datenbanken o. dgl.

10. Verwendung von konfigurierbaren Logikschaltungs-Arrays mit verteilten Speichern zur Entwicklung von speziellen Digital-Schaltkreisen durch Populationsdynamik von interaktiven Teil-Rekonfigurationen von Hardware und Berechnungen, basierend auf genetischer Manipulation von lokale Konfigurationen repräsentierenden Datenstrings.

Hierzu 13 Seite(n) Zeichnungen

– Leerseite –

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

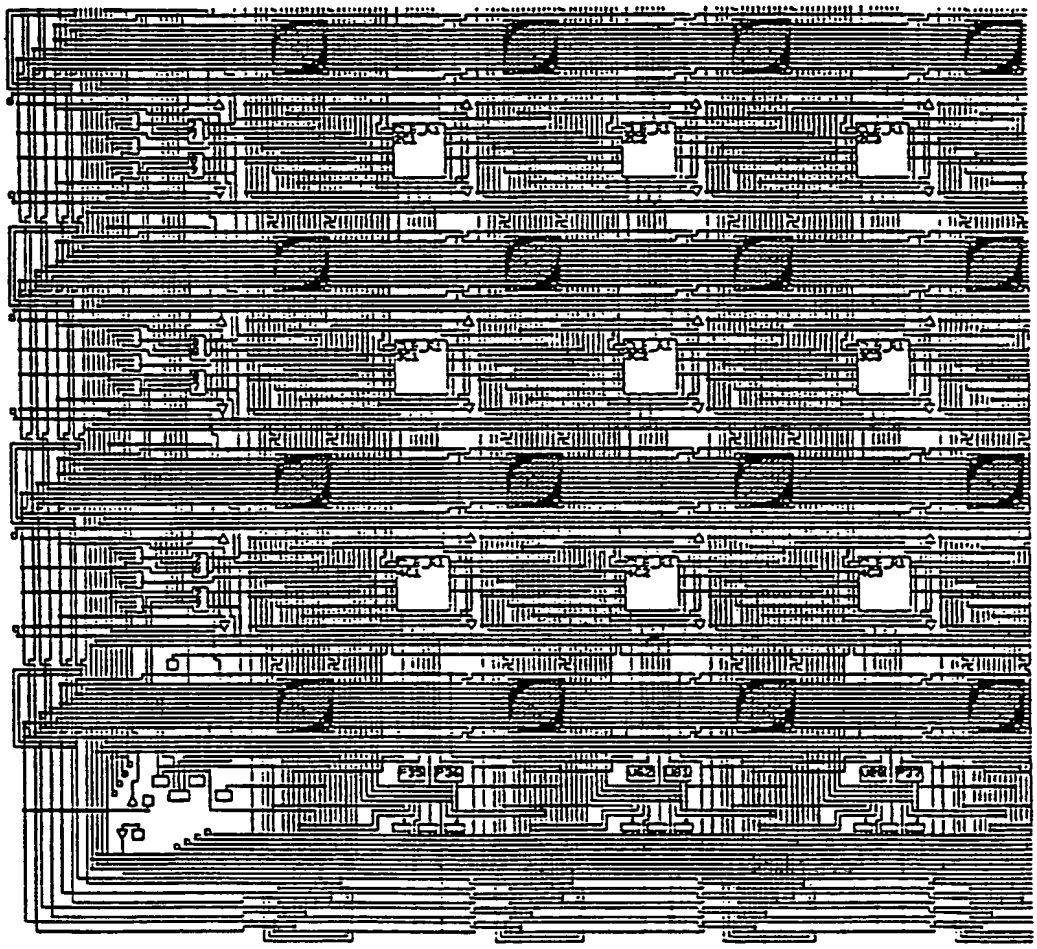


Fig 1

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

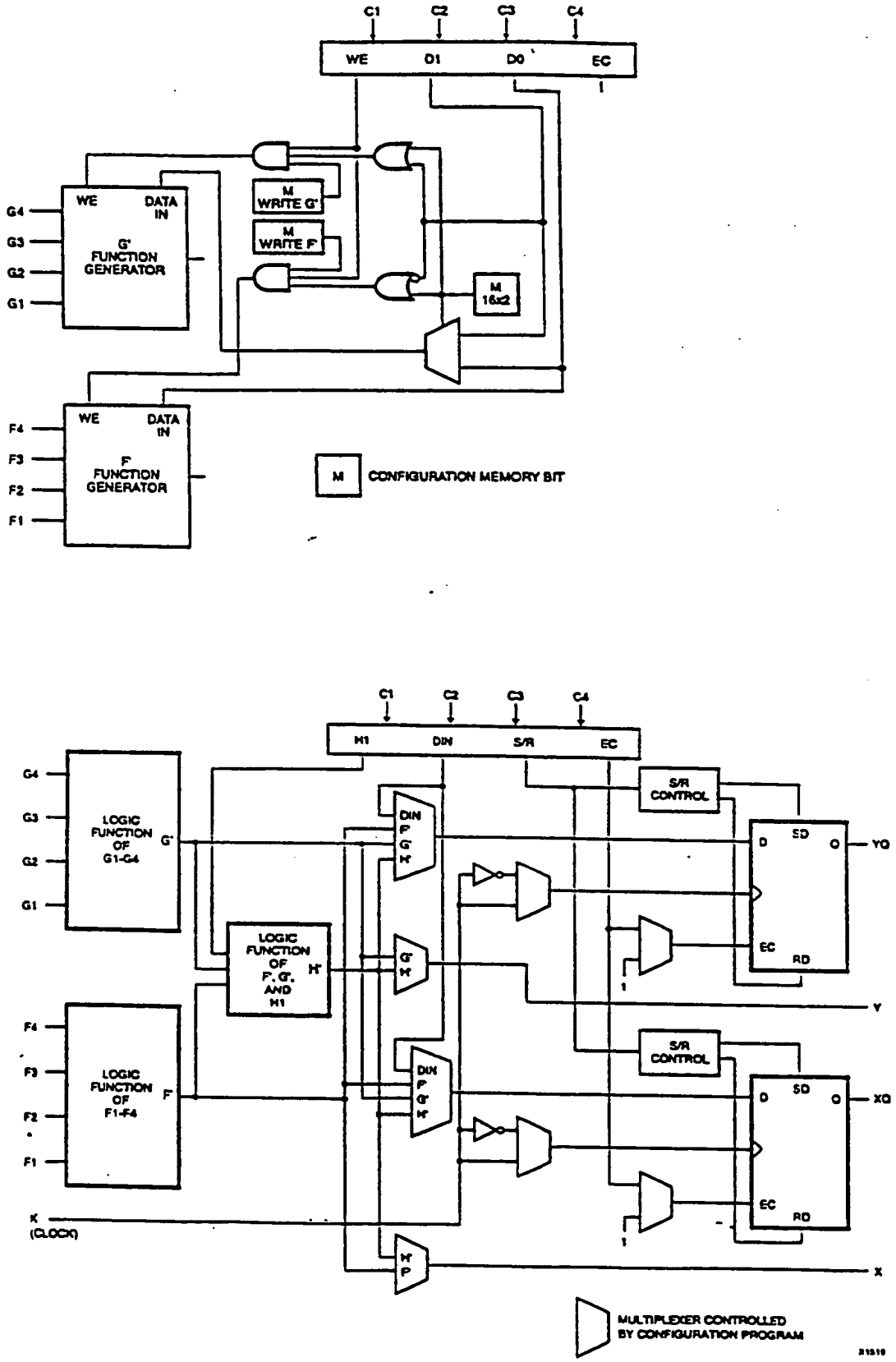


Fig 1

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

Unit Structure and IO for Genetic Processor.

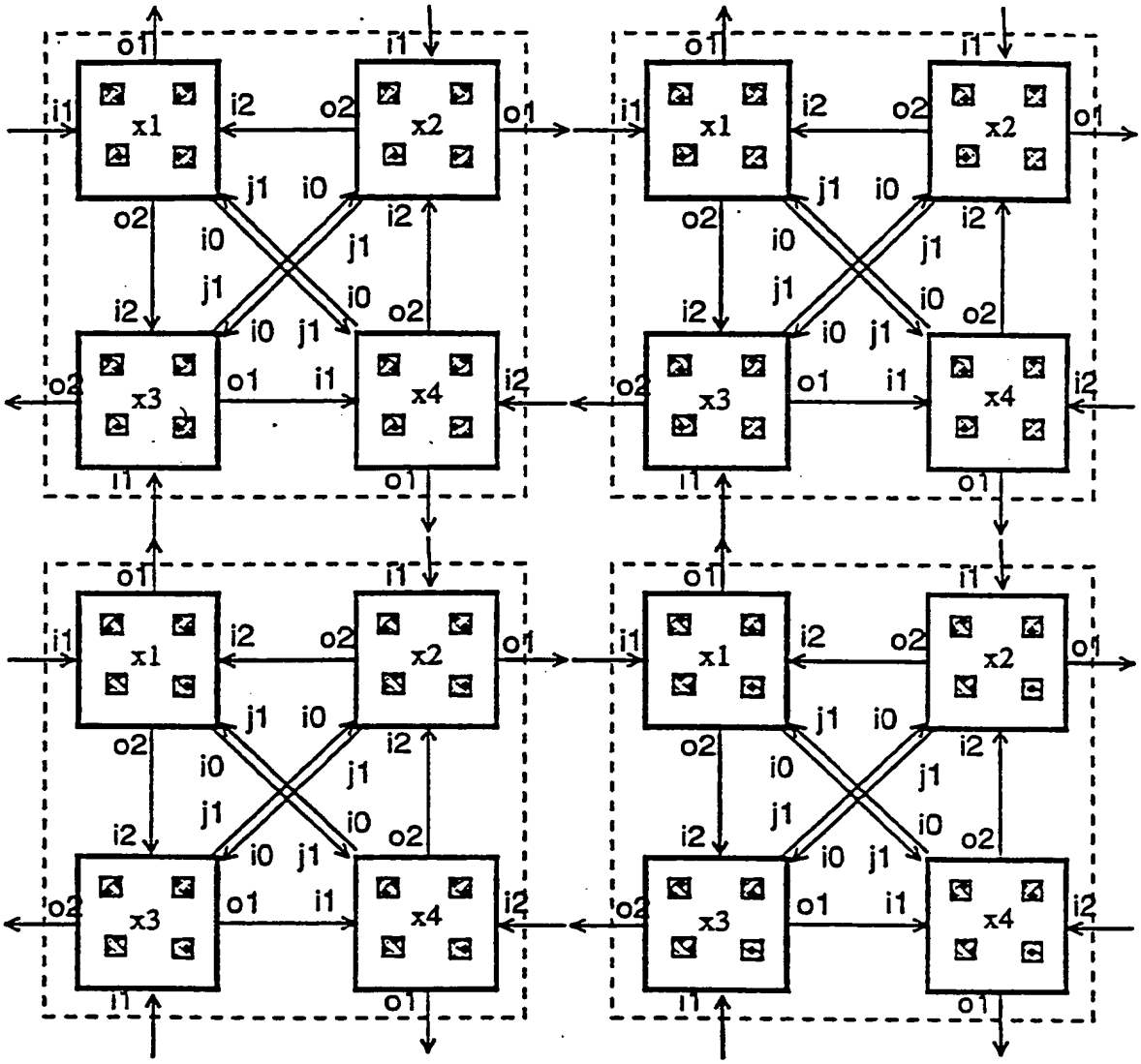
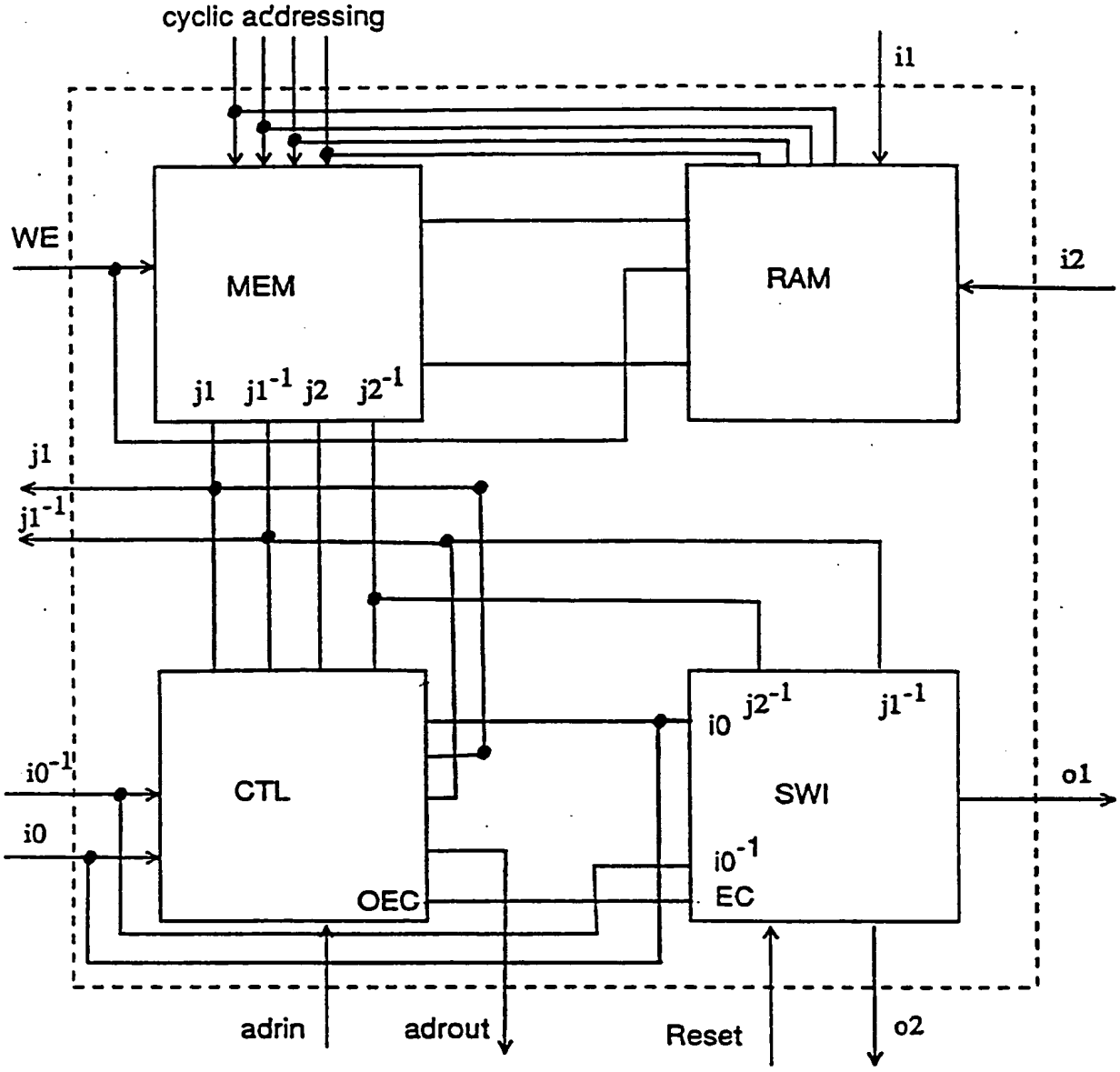


Fig 3

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

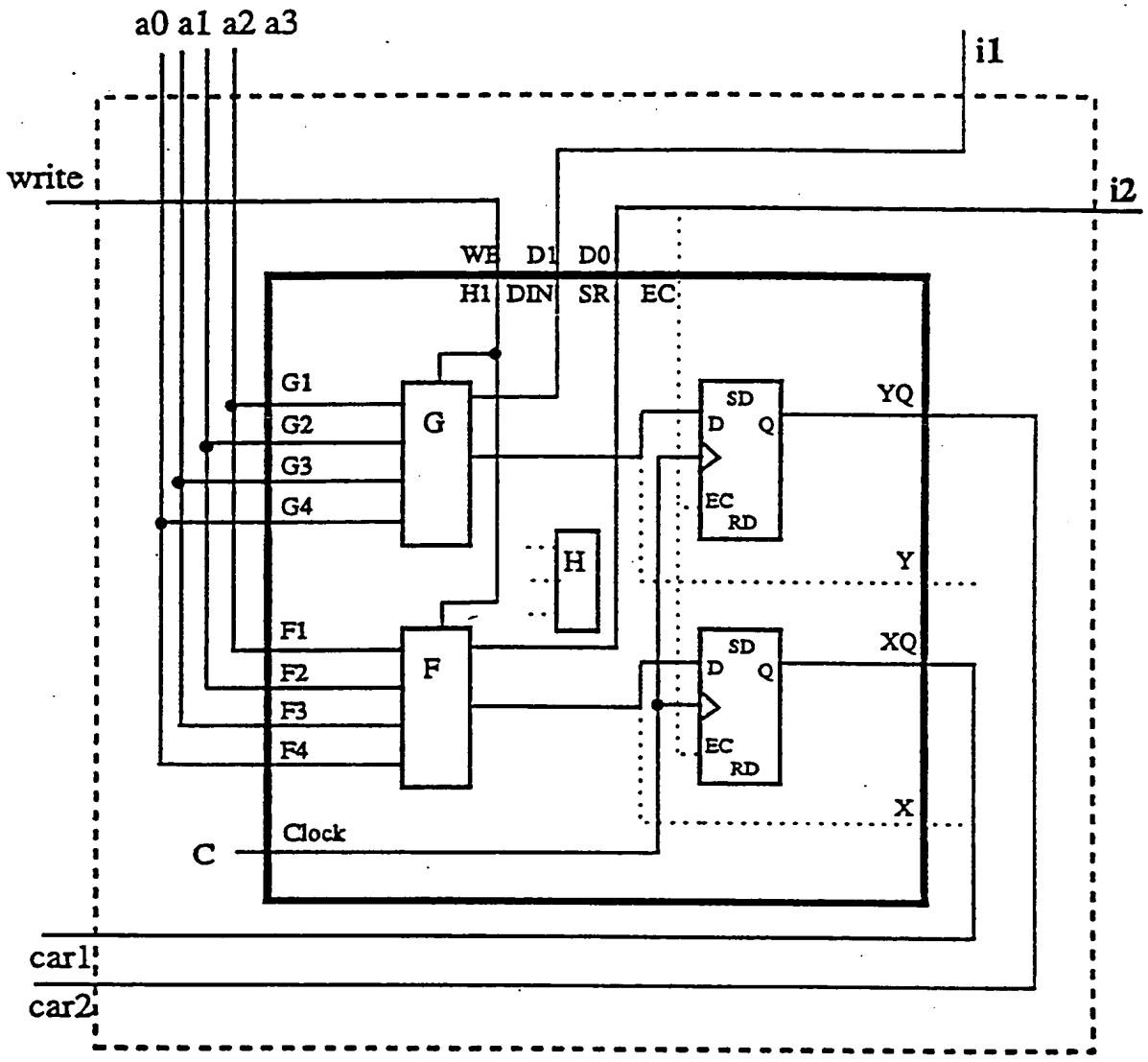


4 CLB Prototype Genetic Switch

Fig 4

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

CLB RAM



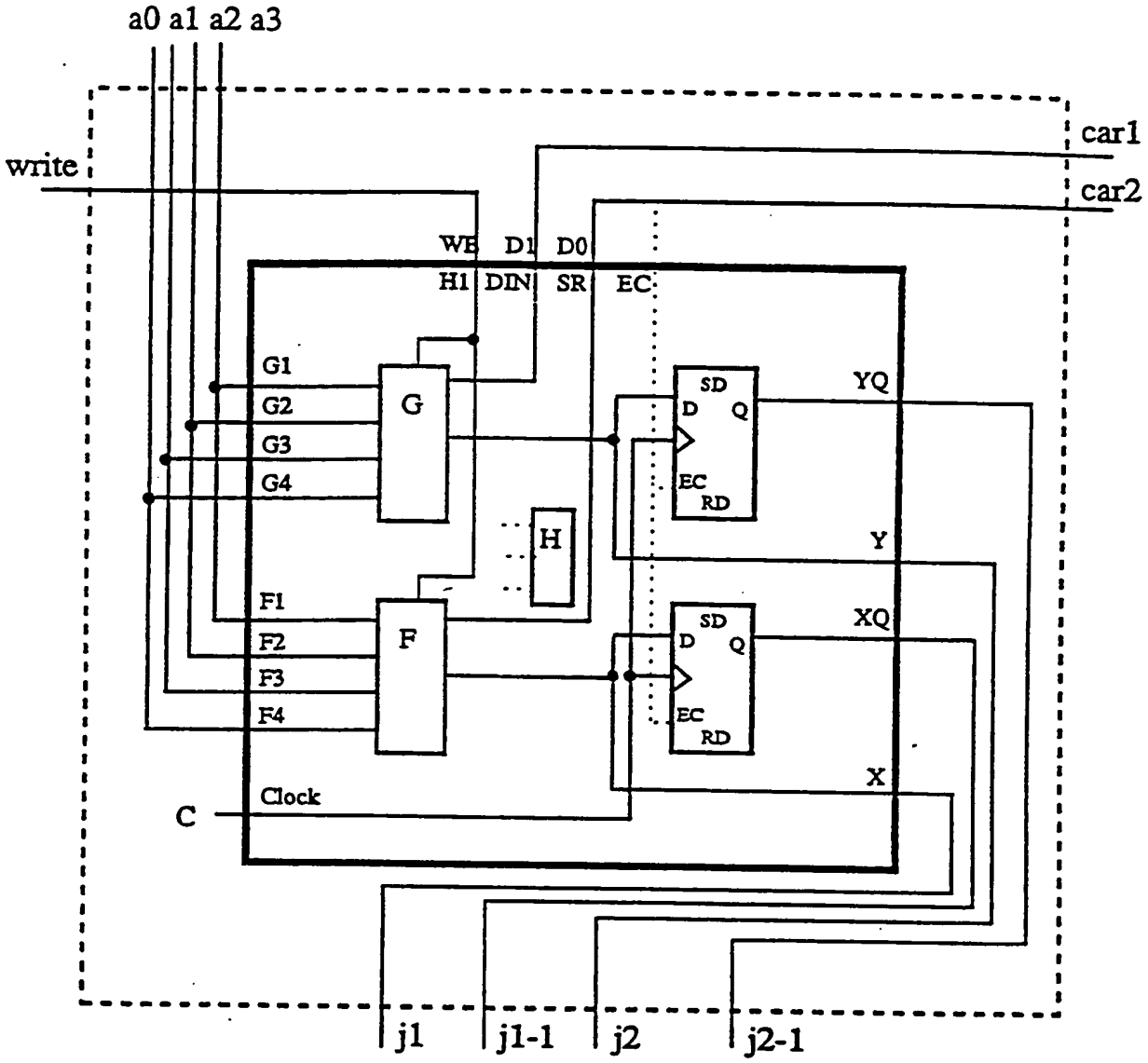
F = RAM

G = RAM

Fig 5A

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

CLB MEM



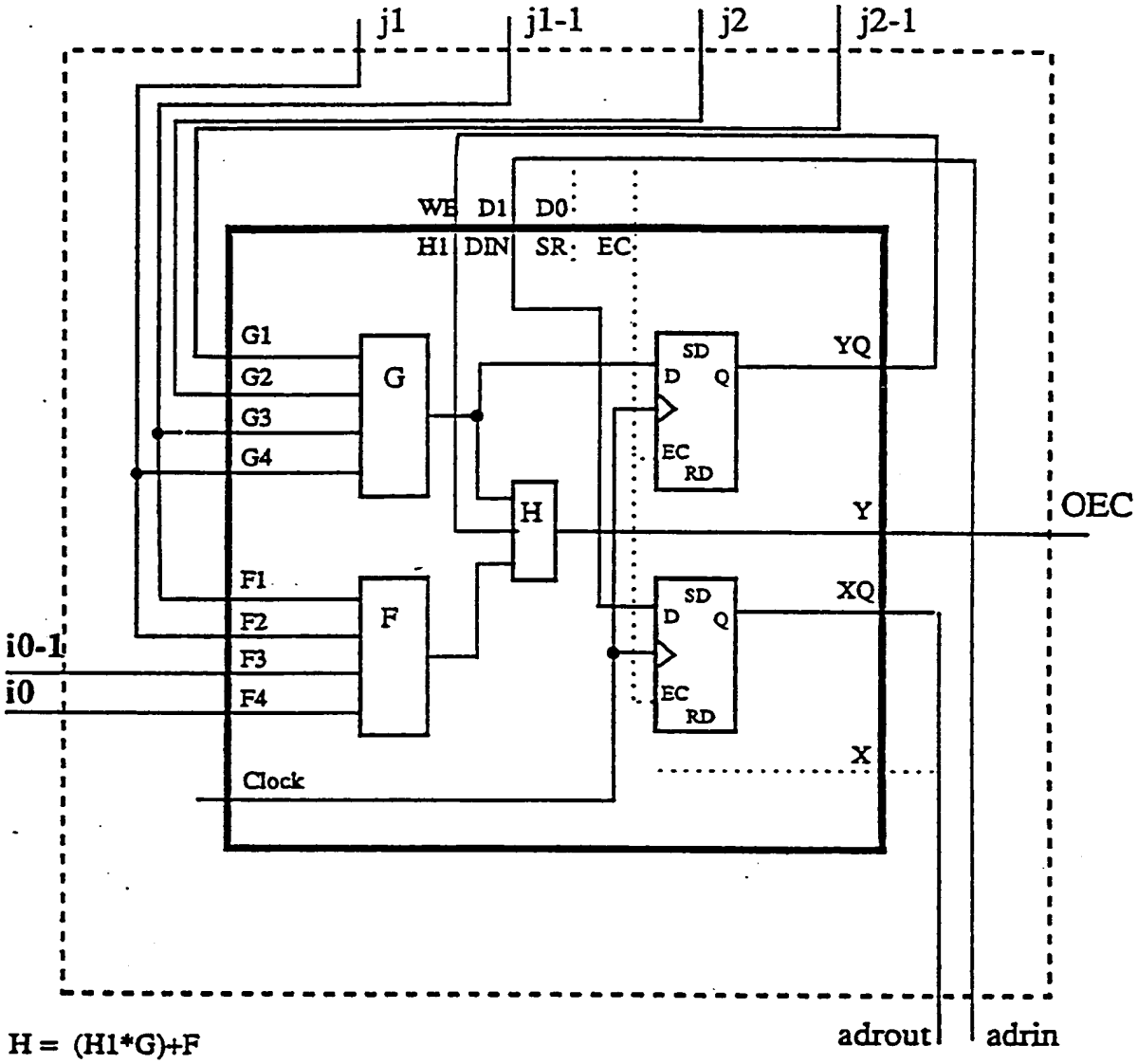
F = RAM

G = RAM

Fig 5B

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

CLB CTL



H = (H1*G)+F

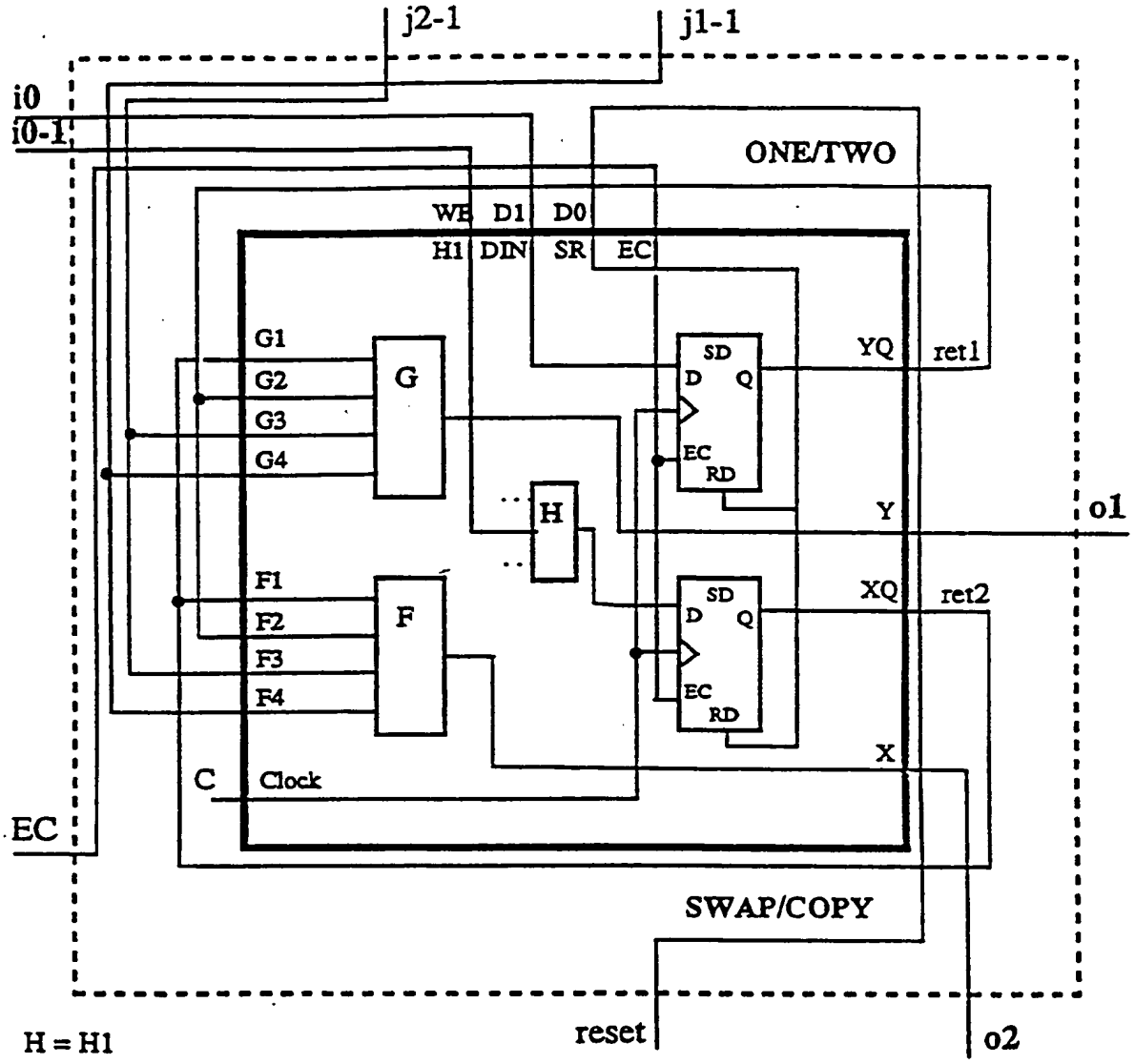
F = ~(j1+j1-1+j2+j2-1)

G = (j1=i0)*(j1-1=i0-1)

Fig 5C

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

CLB SWI



F = ~ret2*(ret1*j2-1+~ret1*j1-1)+ret2*(ret1*j1-1+~ret1*j2-1)

G = ~ret2*(ret1*j1-1+~ret1*j2-1)+ret2*(ret1*j1-1+~ret1*j2-1)

Fig 5D

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

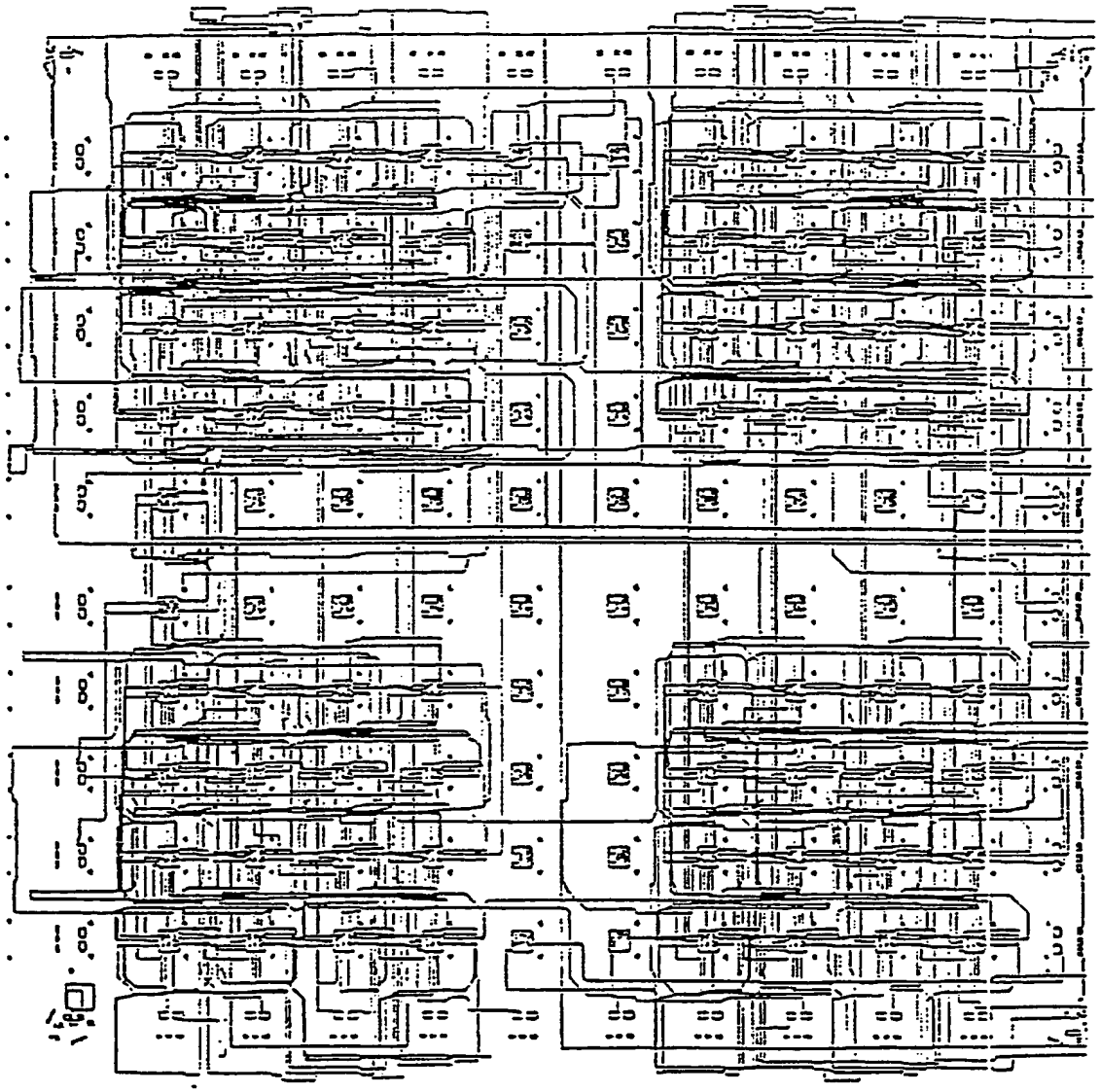


Fig 6

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

GENETIC PROCESSOR I

DEMONSTRATION BOARD LAYOUT

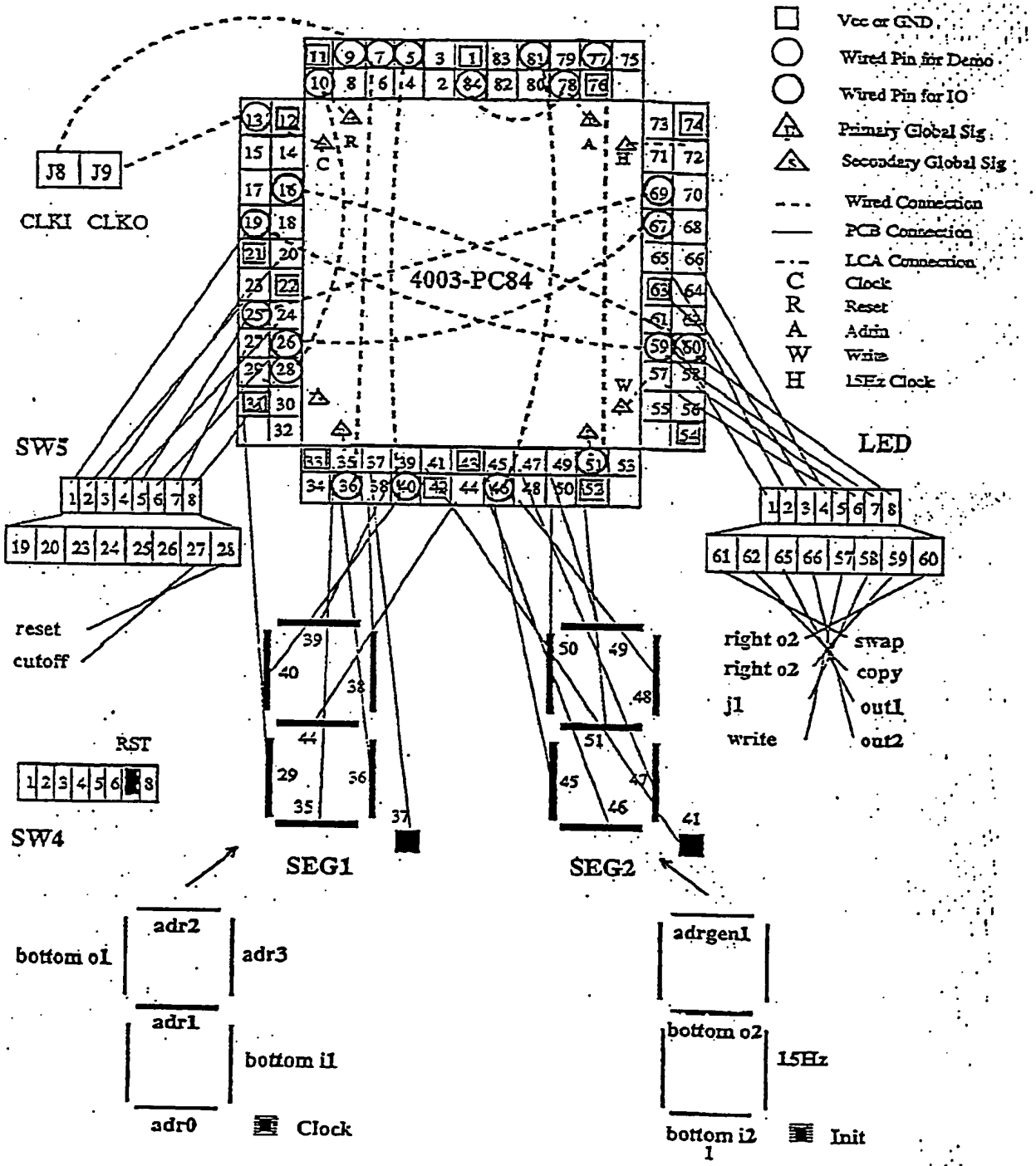


Fig. 7

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

Flow Reactor Interface to Host Computer.

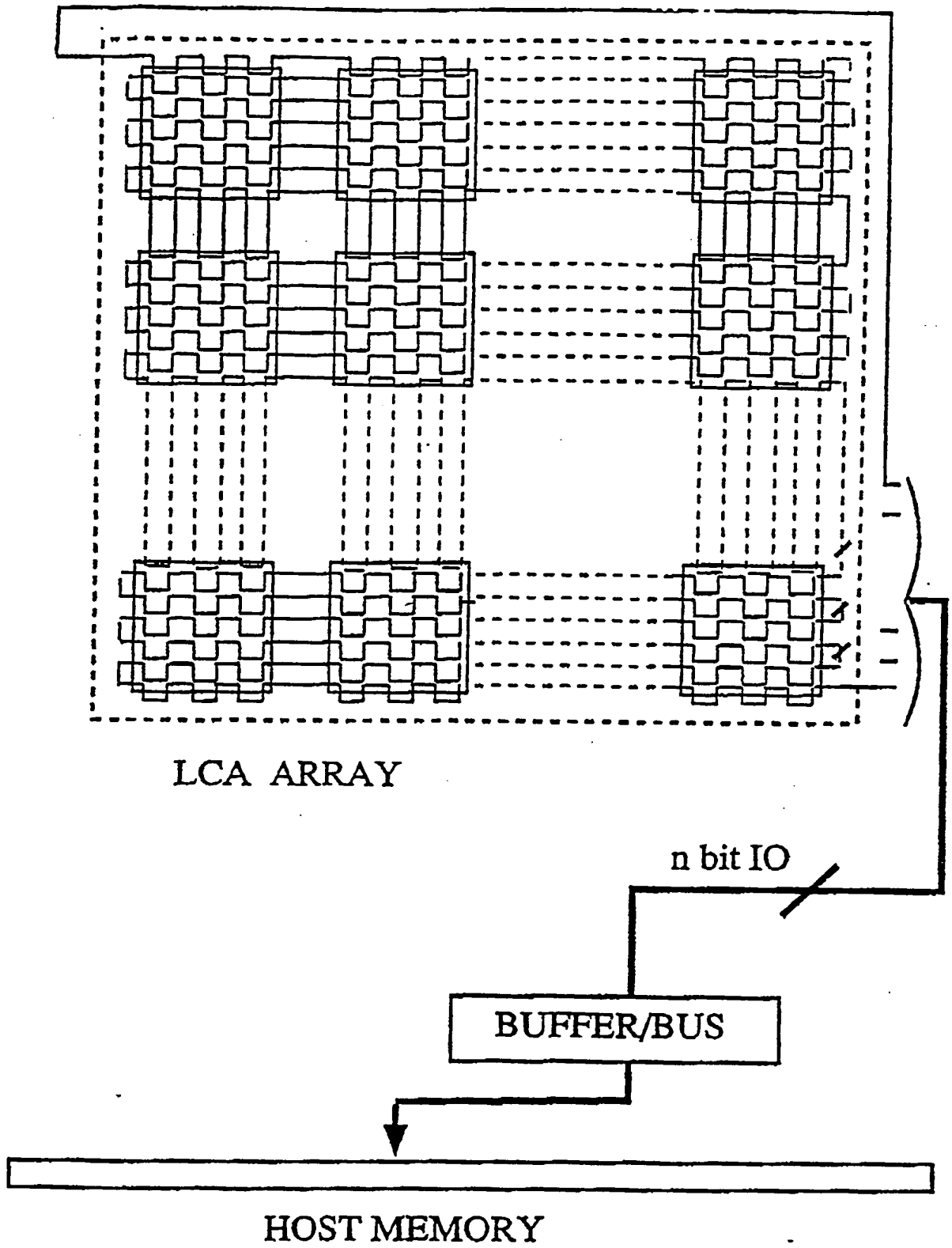
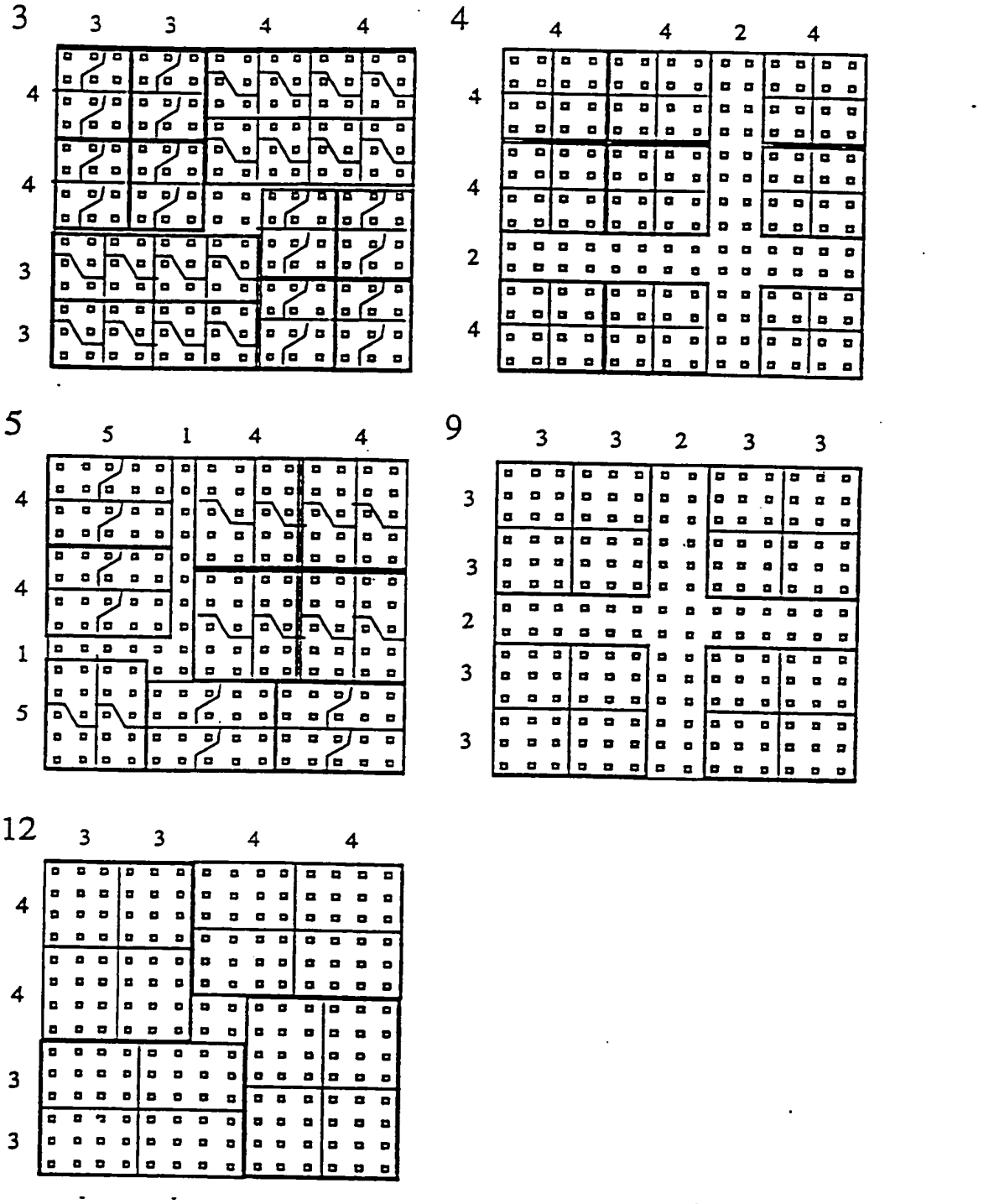


Fig 8

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994



Unit Packing for 4005 LCA (14*14 CLBs)

Fig 9

Nummer: DE 43 35 690 A1
Int. Cl.[5]: G 06 F 15/80
Offenlegungstag: 4. August 1994

ADDRESS GENERATOR WITH INSERTION/DELETION CONTROL

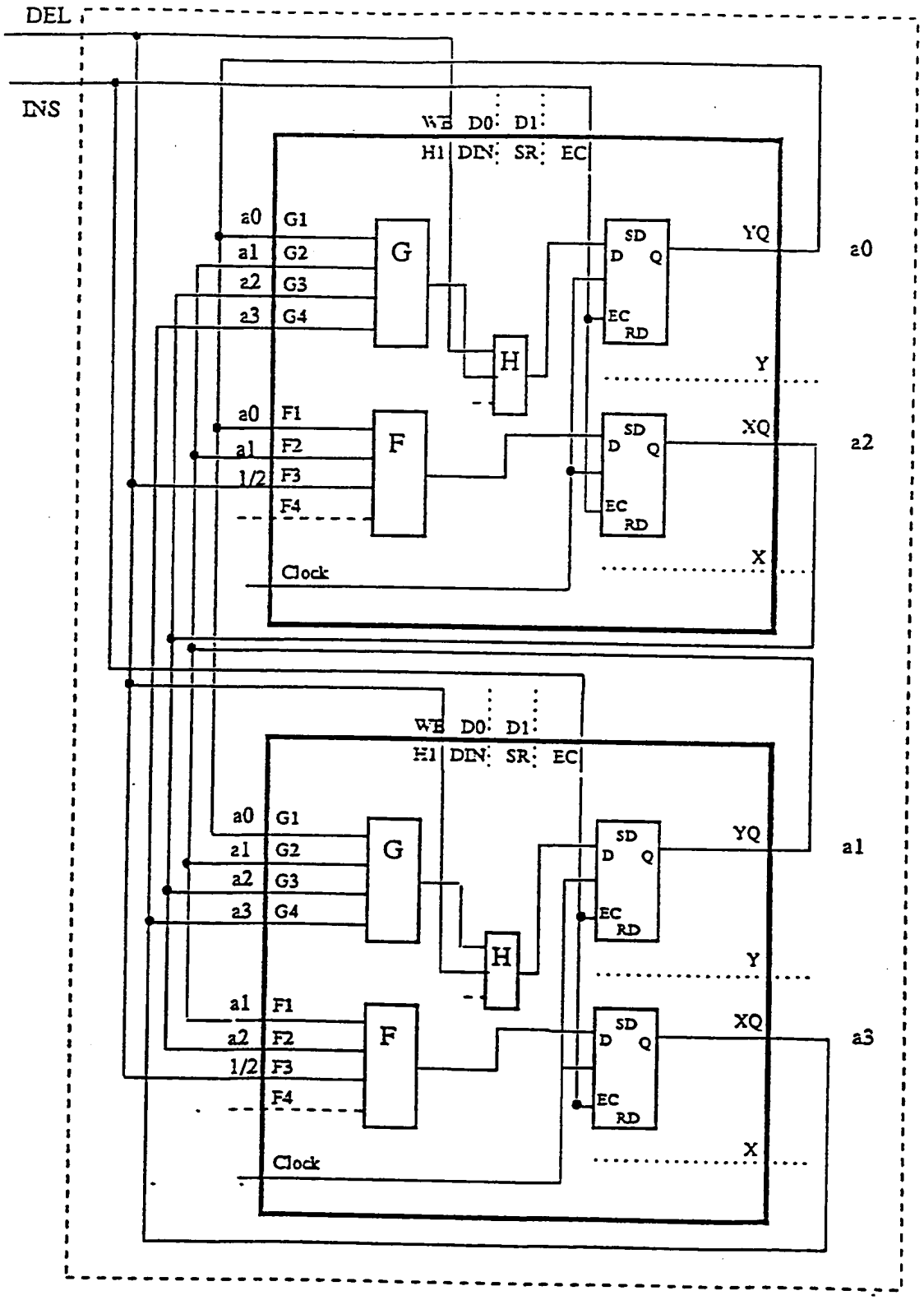


Fig 10